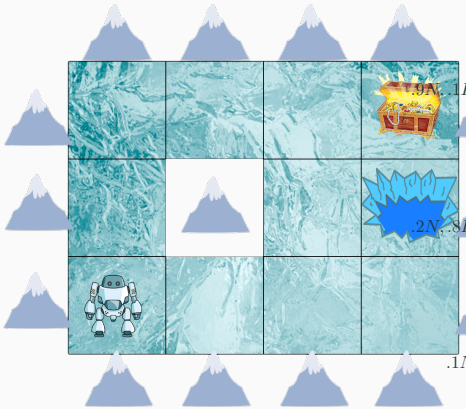
COSC343/AIML402

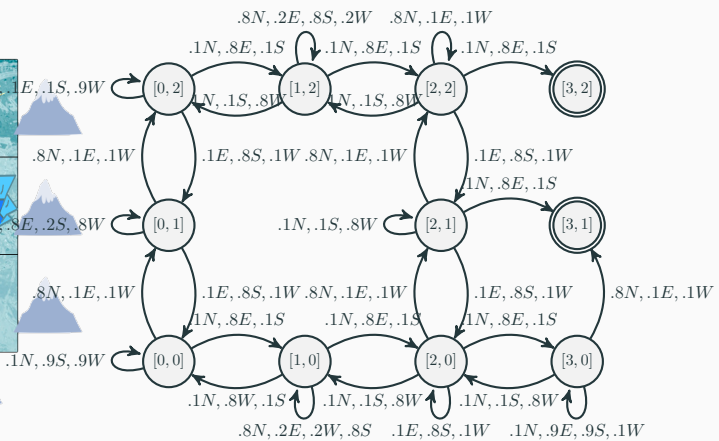
MODULE: MACHINE LEARNING

LECTURE 11&12: REINFORCEMENT LEARNING I & II

Lech Szymanski

Guiding problem: Grid-world





Review: Machine learning types of feedback

Supervised learning	Unsupervised learning	Reinforcement learning
The agent is shown appropriate actions for each instance of percepts	The agent **is not** shown appropriate outputs, just percepts	The agent gets a reward after a sequence of actions
Given table of input/output correspondences, learn a generic rule that governs the table, so that the agent can produce appropriate output for never seen before input	Find patters in data for the purpose of encoding, compressions, generation of similar data, categorisation...	Act in the environment and learn behaviours that bring rewards

Sequential decision problems

Sequential decision problems are a general framework for a set of problems that encompass search and planning as a special case.

Environment is composed of a set of distinct states, and it is assumed to be non-deterministic (i.e. it's stochastic). This means that there is no guarantee of the same outcome if taking the same action in the same state. This is more akin to real life situations.

Formally, a generic sequential decision problem consist of the following elements:

- Agent's **initial state**, s_0
- A set of possible **actions** $\{a_1, a_2, \ldots\}$ when in state s
- A **transition model** which returns a probability distribution over possible states that results from taking action a when in state s; $P(s'|s, a)$
- A **reward** associated with each state (may be zero or negative).

Reinforcement learning

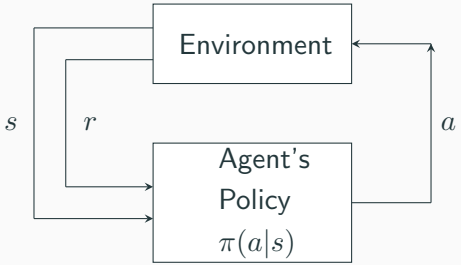
Reinforcement learning (RL) is a set of methods that allow the agent to learn what to do in an unknown (no knowledge of the transition model) stochastic environment. The agent has to deduce the optimal **policy** (what to do in a given sate) from the rewards it manages to receive.

- Agent's **initial state**, s_0
- A set of possible **actions** $\{a_1, a_2, \ldots\}$ when in state s
- ~~A **transition model** which returns a probability distribution over possible states that results from taking action a when in state s; $P(s'|s,a)$~~
- A **reward** associated with each state (may be zero or negative).

The term **reinforcement** relates to the feedback given through the reward at the end of a sequence of steps, which informs the agent whether the final outcome was positive or negative, but not necessarily which states in the sequence relate to this outcome.

RL terminology

The **state**, s_t is the state of environment **as perceived by the agent**. Thus, the agent doesn't know the true state, but associates its percepts with the state.



Policy is a parametrised function, typically labelled $\pi(a|s)$ (the parameters are implicit in this notation), which outputs the probability distribution over possible actions to take in state s; changing the value of parameters of $\pi(s, a)$ changes the the policy for given s.

Reward r is a signal received from the environment quantifying the value (positive is good, negative is bad, zero is neither) of being in state s.

Action a is the action taken by the agent.

Utility

The objective is for the agent to learn the policy that will lead it to navigate through the environment into the states with maximum positive rewards.

The subjective value agent places on being in a given state is referred to as **utility**.

The utility of being in any a state is the reward of that state plus rewards for whatever future states might follow (discounted by $0 < \gamma \leq 1$).

$$\begin{aligned} U(s_t) &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} .... \\ &= r_t + \gamma \Big(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} ... \Big) \\ &= r_t + \gamma U(s_{t+1}) \end{aligned} \tag{1}$$

Note! The utility of being in a **terminal state** (that ends the simulation/game/run) ends up being just the reward of that state, $U(s_t) = r_t$.

Q-value

The **Q-value** Q(s,a) is the expected utility when taking action a in state s.

Q-table is the table of Q-values for all possible combinations of actions and states.

	a_1	a_2	$\dots$	a_K
s_1	$Q(s_1, a_1)$	$Q(s_1, a_2)$	$\dots$	$Q(s_1, a_K)$
s_2	$Q(s_2, a_1)$	$Q(s_2, a_2)$	$\dots$	$Q(s_2, a_K)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
s_N	$Q(s_N, a_1)$	$Q(s_N, a_2)$	$\dots$	$Q(s_N, a_K)$

Q-table can be used as the policy of the agent. Given state s, the best action to take is the one which has the largest Q-value in the row s of the Q-table.

$$U(s) = \max\Big\{Q(s, a_1), \dots, Q(s, a_K)\Big\}$$

Q-value updates

Let's modify Equation 1 to derive $Q(s, a)$ from utility:

$$\begin{aligned} Q(s, a) &= r + \gamma U(s') \\ &= r + \gamma \text{max}\Big\{Q(s', a_1), \ldots, Q(s', a_K)\Big\}, \end{aligned}$$

where s' is the next state after s and r is the reward for being in state s.

The Q-value of action a while in state s is the current reward r plus the Q value of the best possible action that can be taken in s', the state resulting from taking action a in s.

Formalising Q-table based policy

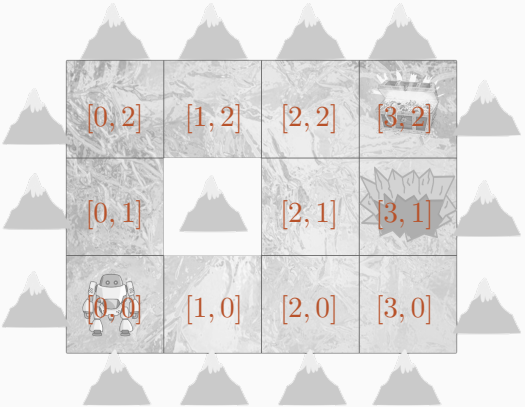
Given the agent with K actions $a \in \{a_1, \ldots, a_K\}$ and function/table Q(s,a):

1. When in state s, choose a_k where $k = \arg\max_k Q(s, a_k)$.

	a_1	a_2	$\ldots$	a_K
s_1	$Q(s_1, a_1)$	$Q(s_1, a_2)$	$\ldots$	$Q(s_1, a_K)$
s_2	$Q(s_2, a_1)$	$Q(s_2, a_2)$	$\ldots$	$Q(s_2, a_K)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
s_N	$Q(s_N, a_1)$	$Q(s_N, a_2)$	$\ldots$	$Q(s_N, a_K)$

Example: Grid-world (Q-table)

	$a = N$	$a = E$	$a = S$	$a = W$
$[0, 0]$	0.519	0.441	0.459	0.473
$[1, 0]$	0.410	0.394	0.416	0.462
$[2, 0]$	0.431	0.340	0.385	0.412
$[3, 0]$	-0.691	-0.159	0.343	0.377
$[0, 1]$	0.590	0.543	0.487	0.543
$[2, 1]$	0.298	-0.558	0.279	0.490
$[3, 1]$	-1.000	-1.000	-1.000	-1.000
$[0, 2]$	0.616	0.673	0.559	0.602
$[1, 2]$	0.673	0.751	0.690	0.627
$[2, 2]$	0.757	0.836	0.528	0.687
$[3, 2]$	1.000	1.000	1.000	1.000



Exploration vs. exploitation

In RL, the agent deduces the policy by trial and error. The only way to find out whether a sequence of actions will lead to a positive or a negative outcome is for the agent to *try something*, see what happens, and then modify the parameters of its policy so as to increase the probability of taking action that leads to positive reward.

The agents needs to **exploit** what is has already learned about the environment in order to make good choices (i.e. go with the best action that the current policy dictates).

The agents needs to **explore** the environment, to find out whether a better strategy exists (i.e. test a new action that wasn't tested in a given situation).

Exploration and exploitation using Q-table-based policy

	a_1	...	a_K
s_1	$Q(s_1, a_1)$	...	$Q(s_1, a_K)$
s_2	$Q(s_2, a_1)$	...	$Q(s_2, a_K)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
s_N	$Q(s_N, a_1)$	...	$Q(s_N, a_K)$

A row of Q-table can be transformed into probability distribution of actions given state using a parameterised softmax function over all the entries of a given row:

$$\pi(a_j|s) = \frac{e^{Q(s,a_j)/T}}{\sum_{k=1}^{K} e^{Q(s,a_j)/T}},$$

where $T > 0$ is a temperature parameter.

The action chosen is the one sampled from distribution $\pi(a_j|s)$.

The larger the T, the more uniform $\pi(a_j|s)$, and the more explorative the agent.

The smaller the T, the bigger the gap between the largest $\pi(a_j|s)$ and other actions, and thus the more exploitative the agent.

Formalising the learning of Q-table based policy

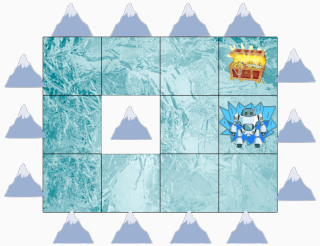
Choose $T > 0$ (for desired degree of exploration) and $0 < \gamma \leq 1$ for desired discount of future rewards

1. Set all values of Q-table to zeros
2. For number of episodes:
3. Reset environment to the initial state $s = s_0, r = r_0$
4. For step $t = 0$ up to max number of steps per episode:
5. If s is a terminal state:
6. Set $Q(s, a_j) = r$ for all $j = 1, \ldots, K$ and break out of for loop
7. Compute $\pi(a_j|s) = \frac{e^{Q(s,a_j)/T}}{\sum_{k=1}^{K} e^{Q(s,a_j)/T}}$
8. Sample $\pi(a_j|s)$ to choose action to take a
9. Take action a, observe next state s' and get reward r'
10. Set $Q(s, a) := r + \gamma \text{max}\Big\{Q(s', a_1), \ldots, Q(s', a_K)\Big\}$
11. Set $s = s'$, $r = r'$

Example: Grid-world Q-based policy learning ($\gamma = 1.0$)

Q-table

	N	E	S	W
(0,0)	0.00	0.00	0.00	0.00
(1,0)	0.00	0.00	0.00	0.00
(2,0)	0.00	0.00	0.00	0.00
(3,0)	0.00	0.00	0.00	0.00
(0,1)	0.00	0.00	0.00	0.00
(2,1)	0.00	0.00	0.00	0.00
s_5 (3,1)	-1.00	-1.00	-1.00	-1.00
(0,2)	0.00	0.00	0.00	0.00
(1,2)	0.00	0.00	0.00	0.00
(2,2)	0.00	0.00	0.00	0.00
(3,2)	0.00	0.00	0.00	0.00



$T = 0.1, \gamma = 1.0$

$p(a_4|s_4) = \text{smax}\big(Q(s_4,:)\big) = [0.25, 0.25, 0.25, 0.25]$

$Q(s_5, a_5) = r_5$

$Q\big((3,1),:\big) = -1.0$

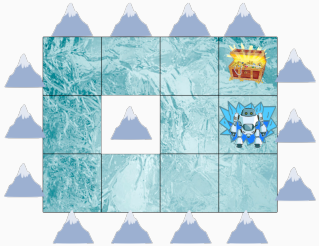
Episode 1:

t	0	1	2	3	4	5				
r_t	0.00	0.00	0.00	0.00	0.00	-1.00				
s_t	(0,0)	(1,0)	(1,0)	(2,0)	(2,1)	(3,1)				
a_t	E	N	N	N	E					
$Q(s_t, a_t)$	0.00	0.00	0.00	0.00	0.00					

Example: Grid-world Q-based policy learning ($\gamma = 1.0$)

Q-table

		N	E	S	W
	(0,0)	0.00	0.00	0.00	0.00
	(1,0)	0.00	0.00	0.00	0.00
	(2,0)	0.00	0.00	0.00	0.00
s_8	(3,0)	0.00	-1.00	0.00	0.00
	(0,1)	0.00	0.00	0.00	0.00
	(2,1)	0.00	0.00	0.00	0.00
s_9	(3,1)	-1.00	-1.00	-1.00	-1.00
	(0,2)	0.00	0.00	0.00	0.00
	(1,2)	0.00	0.00	0.00	0.00
	(2,2)	0.00	0.00	0.00	0.00
	(3,2)	0.00	0.00	0.00	0.00



$T = 0.1, \gamma = 1.0$

$p(a_8|s_8) = \mathsf{smax}(Q(s_8, :)) = [0.25, 0.25, 0.25, 0.25]$

$Q(s_8, a_8) = r_8 + \gamma \mathsf{max}(Q(s_9, :))$

$Q((3,0), E) = 0.0 + 1.0 \cdot -1.00$

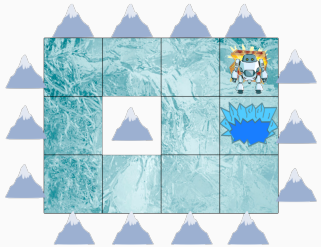
Episode 25:

t	0	1	2	3	4	5	6	7	8	9
r_t	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-1.00
s_t	(0,0)	(0,0)	(1,0)	(1,0)	(1,0)	(1,0)	(2,0)	(2,0)	(3,0)	(3,1)
a_t	W	E	W	S	N	E	S	E	E	
$Q(s_t, a_t)$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	

Example: Grid-world Q-based policy learning ($\gamma = 1.0$)

Q-table

	N	E	S	W
(0,0)	0.00	0.00	0.00	0.00
(1,0)	0.00	0.00	0.00	0.00
(2,0)	0.00	0.00	0.00	0.00
(3,0)	0.00	-1.00	0.00	0.00
(0,1)	0.00	0.00	0.00	0.00
(2,1)	0.00	0.00	0.00	0.00
(3,1)	-1.00	-1.00	-1.00	-1.00
(0,2)	0.00	0.00	0.00	0.00
(1,2)	0.00	0.00	0.00	0.00
(2,2)	0.00	0.00	0.00	0.00
s_6 (3,2)	1.00	1.00	1.00	1.00



$T = 0.1, \gamma = 1.0$

$p(a_5|s_5) = \text{smax}(Q(s_5, :)) = [0.25, 0.25, 0.25, 0.25]$

$Q(s_6, a_6) = r_6$

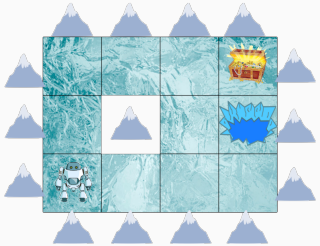
$Q((3,2), :) = 1.0$

Episode 32:

t	0	1	2	3	4	5	6			
r_t	0.00	0.00	0.00	0.00	0.00	0.00	1.00			
s_t	(0,0)	(1,0)	(2,0)	(2,0)	(2,1)	(2,2)	(3,2)			
a_t	E	E	S	N	N	E				
$Q(s_t, a_t)$	0.00	0.00	0.00	0.00	0.00	0.00				

Example: Grid-world Q-based policy learning ($\gamma = 1.0$)

Q-table		N	E	S	W
s_9	(0,0)	0.00	0.00	0.00	1.00
	(1,0)	0.00	0.00	0.00	0.00
	(2,0)	0.00	0.00	0.00	0.00
	(3,0)	-1.00	-1.00	0.00	-1.00
s_8	(0,1)	0.00	0.00	1.00	0.00
	(2,1)	-1.00	-1.00	0.00	0.00
	(3,1)	-1.00	-1.00	-1.00	-1.00
	(0,2)	0.00	0.00	0.00	0.00
	(1,2)	0.00	0.00	0.00	0.00
	(2,2)	0.00	1.00	0.00	0.00
	(3,2)	1.00	1.00	1.00	1.00



$T = 0.1, \gamma = 1.0$

$p(a_8|s_8) = \mathsf{smax}(Q(s_8, :)) = [0.00, 0.00, 1.00, 0.00]$

$Q(s_8, a_8) = r_8 + \gamma \mathsf{max}(Q(s_9, :))$

$Q((0,1), S) = 0.0 + 1.0 \cdot 1.00$

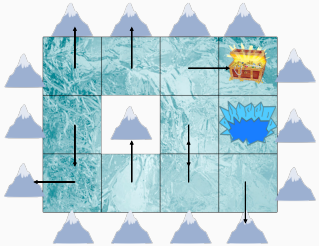
Episode 500:

t	0	1	2	3	4	5	6	7	8	9
r_t	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
s_t	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,1)	(0,0)
a_t	W	W	W	W	W	W	W	W	S	
$Q(s_t, a_t)$	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	

Example: Grid-world Q-based policy learning ($\gamma = 1.0$)

Q-table

	N	E	S	W
(0,0)	0.00	0.00	0.00	1.00
(1,0)	0.00	0.00	0.00	0.00
(2,0)	0.00	0.00	0.00	0.00
(3,0)	-1.00	-1.00	0.00	-1.00
(0,1)	0.00	0.00	1.00	0.00
(2,1)	-1.00	-1.00	0.00	0.00
(3,1)	-1.00	-1.00	-1.00	-1.00
(0,2)	0.00	0.00	0.00	0.00
(1,2)	0.00	0.00	0.00	0.00
(2,2)	0.00	1.00	0.00	0.00
(3,2)	1.00	1.00	1.00	1.00



Temporal difference learning

While learning the policy, $U(s) = \gamma \text{max}\Big\{Q(s', a_1), \ldots, Q(s', a_K)\Big\}$ can swing wildly, especially given the stochastic nature of the environment (i.e. in grid-world, agent doesn't know if s' is the results of action a or slipping). Thus, setting

$$Q(s, a) := r + \gamma \text{max}\Big\{Q(s', a_1), \ldots, Q(s', a_K)\Big\}$$

will make the policy very unstable during the learning process.

In **temporal difference learning** we set a desired $Q^*(s, a)$

$$Q^*(s, a) := r + \gamma \text{max}\Big\{Q(s', a_1), \ldots, Q(s', a_K)\Big\}$$

and move $Q(s, a)$ towards $Q^*(s, a)$ just a bit:

$$Q(s, a) := Q(s, a) + \alpha\Big(Q^*(s, a) - Q(s, a)\Big). \tag{2}$$

Sensible choice of learning parameter $\alpha > 0$ guards $Q(s, a)$ against wild swings.

Formalising temporal difference learning Q-table based policy

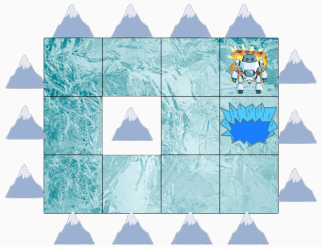
Choose $T > 0$ (for desired degree of exploration), $0 < \gamma \leq 1$ for desired discount of future rewards, and $\alpha > 0$ for desired learning rate:

1. Set all values of Q-table to zeros
2. For number of episodes:
3. Reset environment to the initial state $s = s_0, r = r_0$
4. For step $t = 0$ up to max number of steps per episode:
5. If s is a terminal state:
6. Set $Q(s, a_j) = r$ for all $j = 1, \ldots, K$ and break out of for loop
7. Compute $\pi(a_j|s) = \frac{e^{Q(s,a_j)/T}}{\sum_{k=1}^{K} e^{Q(s,a_j)/T}}$
8. Sample $\pi(a_j|s)$ to choose action to take a
9. Take action a, observe next state s' and get reward r'
10. Set desired $Q^*(s, a) := r + \gamma \text{max}\Big\{Q(s', a_1), \ldots, Q(s', a_K)\Big\}$
11. Set $Q(s, a) := Q(s, a) + \alpha\Big(Q^*(s, a) - Q(s, a)\Big)$
12. Set $s = s'$, $r = r'$

Example: Grid-world Q-based policy temporal difference learning ($\gamma = 1.0$)

Q-table

	N	E	S	W
(0,0)	0.93	0.79	0.83	0.36
(1,0)	0.36	0.80	0.28	0.81
(2,0)	0.85	0.03	0.11	0.11
(3,0)	-0.34	-0.10	0.05	0.37
(0,1)	0.98	0.11	0.17	0.11
(2,1)	0.78	-0.10	0.02	0.01
(3,1)	-1.00	-1.00	-1.00	-1.00
(0,2)	0.13	0.99	0.15	0.03
(1,2)	0.07	0.99	0.00	0.01
s_4 (2,2)	0.19	1.00	0.03	0.02
s_5 (3,2)	1.00	1.00	1.00	1.00



$\alpha = 0.1, T = 0.1, \gamma = 1.0$

$p(a_4|s_4) = \text{smax}\big(Q(s_4,:)\big) = [0.00, 1.00, 0.00, 0.00]$

$Q^*(s_4,a_4) = r_4 + \gamma \text{max}\big(Q(s_5,:)\big) = 0.0 + 1.0 \cdot 1.00 = 1.00$

$Q\big((2,2),E\big) \mathrel{+}= \alpha\Big(Q^*\big((2,2),E\big) - Q\big((2,2),E\big)\Big) \mathrel{+}= 0.1 \cdot \Big(1.00 - 0.99\Big)$

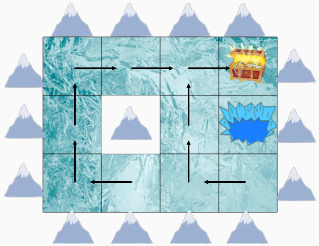
Episode 500:

t	0	1	2	3	4	5				
r_t	0.00	0.00	0.00	0.00	0.00	1.00				
s_t	(0,0)	(0,1)	(0,2)	(1,2)	(2,2)	(3,2)				
a_t	N	N	E	E	E					
$Q(s_t,a_t)$	0.92	0.98	0.99	0.99	0.99					

Example: Grid-world Q-based policy temporal difference learning ($\gamma = 1.0$)

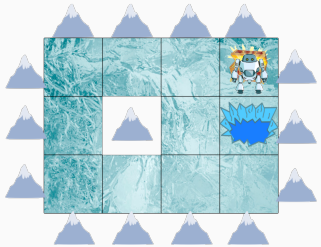
Q-table

	N	E	S	W
(0,0)	0.93	0.79	0.83	0.36
(1,0)	0.36	0.80	0.28	0.81
(2,0)	0.85	0.03	0.11	0.11
(3,0)	-0.34	-0.10	0.05	0.37
(0,1)	0.98	0.11	0.17	0.11
(2,1)	0.78	-0.10	0.02	0.01
(3,1)	-1.00	-1.00	-1.00	-1.00
(0,2)	0.13	0.99	0.15	0.03
(1,2)	0.07	0.99	0.00	0.01
(2,2)	0.19	1.00	0.03	0.02
(3,2)	1.00	1.00	1.00	1.00



Example: Grid-world Q-based policy temporal difference learning ($\gamma = 0.9$)

Q-table		N	E	S	W
	(0,0)	0.47	0.37	0.41	0.43
	(1,0)	0.29	0.10	0.25	0.40
	(2,0)	0.08	0.00	0.06	0.10
	(3,0)	-0.31	-0.02	0.00	-0.00
	(0,1)	0.55	0.16	0.18	0.48
	(2,1)	0.20	-0.17	-0.02	0.16
	(3,1)	-1.00	-1.00	-1.00	-1.00
	(0,2)	0.27	0.64	0.01	0.55
	(1,2)	0.05	0.72	0.01	0.03
s_7	(2,2)	0.09	0.83	0.02	0.02
s_8	(3,2)	1.00	1.00	1.00	1.00



$\alpha = 0.1, T = 0.1, \gamma = 0.9$

$p(a_7|s_7) = \text{smax}\big(Q(s_7,:)\big) = [0.00, 1.00, 0.00, 0.00]$

$Q^*(s_7, a_7) = r_7 + \gamma \text{max}\big(Q(s_8,:)\big) = 0.0 + 0.9 \cdot 1.00 = 0.90$

$Q\big((2,2),E\big) \mathrel{+}= \alpha\Big(Q^*\big((2,2),E\big) - Q\big((2,2),E\big)\Big) \mathrel{+}= 0.1 \cdot \Big(0.90 - 0.82\Big)$

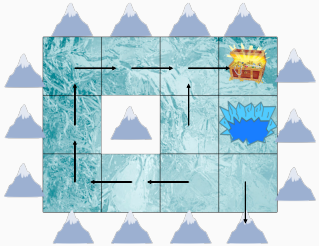
Episode 500:

t	0	1	2	3	4	5	6	7	8	
r_t	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	
s_t	(0,0)	(0,0)	(0,0)	(0,1)	(0,2)	(1,2)	(1,2)	(2,2)	(3,2)	
a_t	W	W	N	N	E	E	E	E		
$Q(s_t, a_t)$	0.43	0.43	0.46	0.55	0.64	0.72	0.71	0.82		

Example: Grid-world Q-based policy temporal difference learning ($\gamma = 0.9$)

Q-table

	N	E	S	W
(0,0)	0.47	0.37	0.41	0.43
(1,0)	0.29	0.10	0.25	0.40
(2,0)	0.08	0.00	0.06	0.10
(3,0)	-0.31	-0.02	0.00	-0.00
(0,1)	0.55	0.16	0.18	0.48
(2,1)	0.20	-0.17	-0.02	0.16
(3,1)	-1.00	-1.00	-1.00	-1.00
(0,2)	0.27	0.64	0.01	0.55
(1,2)	0.05	0.72	0.01	0.03
(2,2)	0.09	0.83	0.02	0.02
(3,2)	1.00	1.00	1.00	1.00



Lab 6: Grid-world

Objectives:

- Test your understanding of RL concepts
- Implement (and get a deeper understanding of) the Q-learning algorithm

Study guide

- Terms and definitions: policy, rewards, utility, q-value, q-table.
- Understand how utility captures current and future (discounted) rewards.
- Know how to use Q-table as agent's policy for choosing the best action in a given state
- Understand how softmax function can be used to derive probability distribution of actions given state from the Q-table (for the purposes of exploration).
- Terms and definitions: exploration, exploitation, temporal difference learning
- Know the temporal difference learning algorithm for Q-table based policy – remembering all the mathematical details it not necessary, just good understanding of the steps involved and how it all works at a high level

Reading for this lecture

Read AIMA Ch.17.1,17.3, Ch.21.1-21.3

What's next?

Decision trees

Read AIMA Ch.19.3