Reinforcement Learning 2 Q&A

Oana Cocarascu & Helen Yannakoudakis

Department of Informatics
King's College London

Pacman coursework

- Q: What are we allowed to modify in the CW? Can we modify the signature of the GameStateFeatures class?

```
class GameStateFeatures:
    """
    Wrapper class around a game state where you can extract
    useful information for your Q-learning algorithm

    WARNING: We will use this class to test your code, but the functionality
    of this class will not be tested itself
    """

    def __init__(self, state: GameState):
        """
        Args:
            state: A given game state object
        """

        "*** YOUR CODE HERE ***"
        util.raiseNotDefined()
```

- Q: Can we modify the `__init__` method of the QLearnAgent?
- Q: Are the "***Your code here***" sections the only ones where we are allowed to write code in?

Pacman coursework

- Q: Are we allowed to use code from `pacman_utils` in the CW?

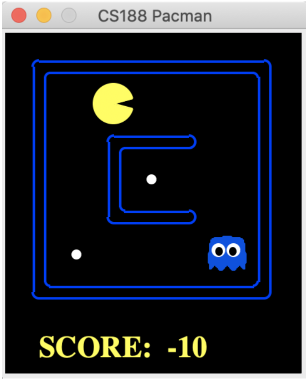
```python
# WARNING: You will be tested on the functionality of this method
# DO NOT change the function signature
def getAction(self, state: GameState) -> Directions:
    """
    Choose an action to take to maximise reward while
    balancing gathering data for learning

    If you wish to use epsilon-greedy exploration, implement it in this method.
    HINT: look at pacman_utils.util.flipCoin

    Args:
        state: the current state

    Returns:
        The action to take
    """
    # The data we have about the state of the game
    legal = state.getLegalPacmanActions()
    if Directions.STOP in legal:
        legal.remove(Directions.STOP)
```

Pacman coursework

- Q: Are we allowed to use the `generateSuccessors` functions in the CW?
- A: No. In RL, you wouldn't have direct access to the successor state.

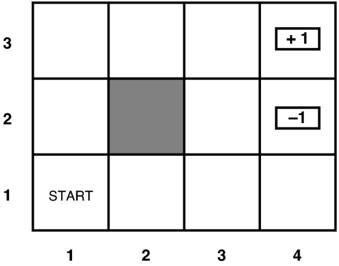
Pacman coursework

- Q: Are we allowed to do ApproximateQLearning for the CW?
- A: Not needed for this map, as the number of states is small.
- For CW2, you will have to implement Q-learning algorithm.

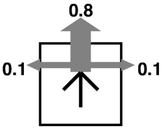


Rewards

- Q: Do we have to set our own rewards for a given state in Q Learning? What happens if we do not know the exact reward?

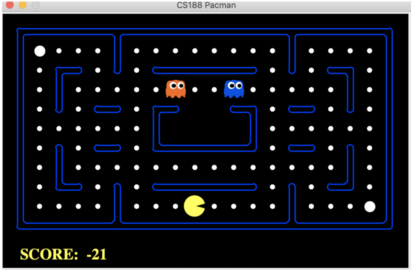


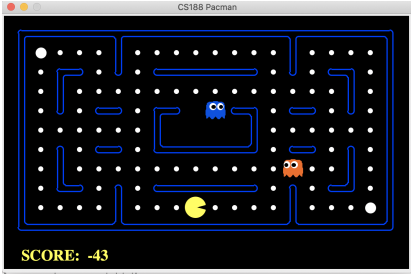
(a)



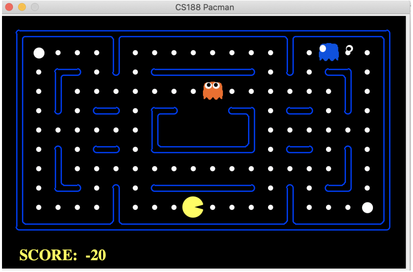
(b)

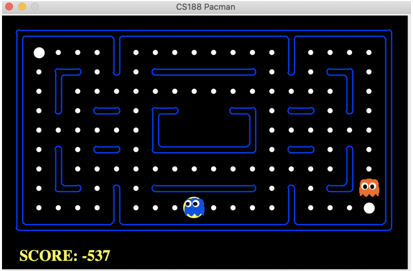
Pacman (when not moving)





Pacman (when losing)



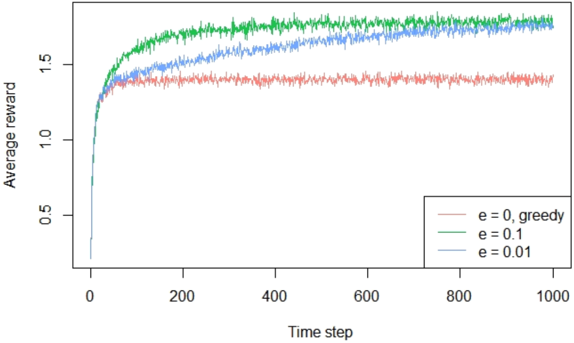


Exploration vs exploitation

- Q: How can agents decide to stop learning? When would they choose to do that?
- Q: Can agents decide when to stop picking actions in an ϵ-Greedy manner and switch to Greedy?

Exploration vs exploitation

- Q: How can agents decide to stop learning? When would they choose to do that?
- Q: Can agents decide when to stop picking actions in an ϵ-Greedy manner and switch to Greedy?



Pacman

- Q: How can agents decide to stop learning? When would they choose to do that?
- Q: Can agents decide when to stop picking actions in an E-Greedy manner and switch to Greedy?

`final()` counts the number of episodes and compares it to `numTraining` — that is how the agent knows when training is over and it is showtime. We set `epsilon` and `alpha` to zero at that point because (as you know from the lecture) these are parameters that control learning. An ϵ-greedy learner chooses not to do the best action that it knows with a probability ϵ, so if we set ϵ to zero the learner will always do what it thinks is best (and it won't get killed because it does something random)[3]. Similarly, if you set α to zero in a Q-learning/SARSA/temporal difference learner, then the update doesn't change the Q-values/utilities.

```
python pacman.py -p QLearnAgent -x 2000 -n 2010 -l smallGrid
```

- For the coursework, train the learner for 2000 episodes and then run it for 10 non-training episodes.

Exploration functions

- ϵ-greedy:

$$a_t = \begin{cases} a_t^* & \text{with probability} \quad 1-\epsilon \\ \textit{random action} & \text{with probability} \quad \epsilon, \qquad \epsilon \ll 1 \end{cases}$$

- Softmax to select non-optimal actions based on their reward:

$$P(a) = \frac{e^{\frac{Q_t(a)}{\tau}}}{\sum_{b=1}^{n} e^{\frac{Q_t(b)}{\tau}}}$$

Exploration functions

- Exploration function $f(u, n)$:

$$f(u,n) = \begin{cases} R^+ & \text{if } n < N_e \\ u & \text{otherwise} \end{cases}$$

  - ADP with exploration function:

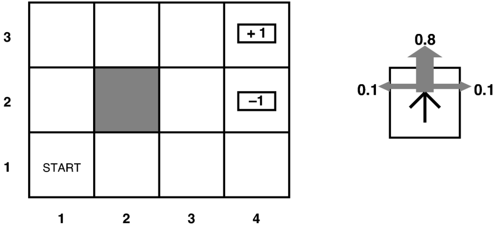
$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} f\left(\sum_{s'} P(s'|s,a) U_i(s'), N(s,a) \right)$$

- Upper-Confidence-Bound action selection:

$$A_t \doteq argmax_a \left[Q_t(a) + c\sqrt{\frac{lnt}{N_t(a)}} \right]$$

- ...

Grid example - Ex1 Tutorial 8

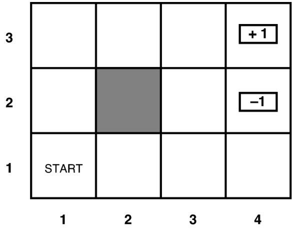


(a) (b)

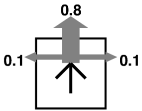
- The Bellman equation for state $(1, 1)$ is:
  $U((1,1)) = R((1,1)) + \gamma \max_{a \in \{Up, Down, Left, Right\}} \sum_{s'} Pr(s'|(1,1), a)U(s')$
- Utility of $(1, 1)$ = reward for being in $(1, 1) +$ the discounted expected utility that will be gained by taking action in $(1, 1)$.
- We compute the expected utility of all the possible actions, and use the maximum expected utility.

Grid example - Ex2 Tutorial 8

- Value iteration for state $(1,1)$, using data from Tutorial 7.



(a)



(b)

- Value iteration uses the Bellman equation as a rule for updating $U(s)$ based on the current estimates of $U(s')$ of neighbouring states:
$U((1,1)) \leftarrow R((1,1)) + \gamma \max_{a \in \{Up, Down, Left, Right\}} \sum_{s'} Pr(s'|(1,1),a)U(s')$

Grid example - Ex2 Tutorial 8

$$
\begin{aligned}
U((1,1)) \leftarrow R((1,1)) \\
+ \gamma \max(&Pr((1,2)|(1,1), Up)U((1,2)) \\
&\quad + Pr((2,1)|(1,1), Up)U((2,1)) \\
&\quad + Pr((1,1)|(1,1), Up)U((1,1)), \\
&Pr((2,1)|(1,1), Right)U((2,1)) \\
&\quad + Pr((1,2)|(1,1), Right)U((1,2)) \\
&\quad + Pr((1,1)|(1,1), Right)U((1,1)), \\
&Pr((1,1)|(1,1), Down)U((1,1)) \\
&\quad + Pr((2,1)|(1,1), Down)U((2,1)) \\
&\quad + Pr((1,1)|(1,1), Down)U((1,1)), \\
&Pr((1,1)|(1,1), Left)U((1,1)) \\
&\quad + Pr((1,2)|(1,1), Left)U((1,2)) \\
&\quad + Pr((1,1)|(1,1), Left)U((1,1))
\end{aligned}
$$

- Some values are 0 because of the fixed policy used to get the estimates.

Grid example - Ex2 Tutorial 8

$$
\begin{aligned}
U((1,1)) \leftarrow R((1,1)) & \\
+\gamma \max(&Pr((1,2)|(1,1),Up)U((1,2)) \\
&\quad + Pr((2,1)|(1,1),Up)U((2,1)) \\
&\quad + Pr((1,1)|(1,1),Up)U((1,1)), \\
&Pr((2,1)|(1,1),Right)U((2,1)) \\
&\quad + Pr((1,2)|(1,1),Right)U((1,2)) \\
&\quad + Pr((1,1)|(1,1),Right)U((1,1)), \\
&Pr((1,1)|(1,1),Down)U((1,1)) \\
&\quad + Pr((2,1)|(1,1),Down)U((2,1)) \\
&\quad + Pr((1,1)|(1,1),Down)U((1,1)), \\
&Pr((1,1)|(1,1),Left)U((1,1)) \\
&\quad + Pr((1,2)|(1,1),Left)U((1,2)) \\
&\quad + Pr((1,1)|(1,1),Left)U((1,1))
\end{aligned}
$$

- If we obtained: $U((1,1)) = 0.71$, $U((1,2)) = 0.75$, $U((3,3)) = 0.912$, and $P((1,2)|(1,1),Up) = 0.75$, $P((1,1)|(1,1),Up) = 0.25$, $P((4,3)|(3,3),Right) = 0.6$, $P((3,2)|(3,3),Right) = 0.4$

Grid example - Ex2 Tutorial 8

$$
\begin{aligned}
U((1,1)) \leftarrow R((1,1)) \\
&+\gamma \max(Pr((1,2)|(1,1),Up)U((1,2)) \\
&\quad + Pr((2,1)|(1,1),Up)U((2,1)) \\
&\quad + Pr((1,1)|(1,1),Up)U((1,1)), \\
&Pr((2,1)|(1,1),Right)U((2,1)) \\
&\quad + Pr((1,2)|(1,1),Right)U((1,2)) \\
&\quad + Pr((1,1)|(1,1),Right)U((1,1)), \\
&Pr((1,1)|(1,1),Down)U((1,1)) \\
&\quad + Pr((2,1)|(1,1),Down)U((2,1)) \\
&\quad + Pr((1,1)|(1,1),Down)U((1,1)), \\
&Pr((1,1)|(1,1),Left)U((1,1)) \\
&\quad + Pr((1,2)|(1,1),Left)U((1,2)) \\
&\quad + Pr((1,1)|(1,1),Left)U((1,1))
\end{aligned}
$$

- If we obtained: $U((1,1)) = 0.71$, $U((1,2)) = 0.75$, $U((3,3)) = 0.912$, and $P((1,2)|(1,1),Up) = 0.75$, $P((1,1)|(1,1),Up) = 0.25$, $P((4,3)|(3,3),Right) = 0.6$, $P((3,2)|(3,3),Right) = 0.4$

Grid example - Ex3 Tutorial 8

- Temporal Difference update for state $(1, 1)$, where $\gamma = 1$ and $\alpha = 0.1$.
- Temporal difference update rule:

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha(R(s) + \gamma U^{\pi}(s') - U^{\pi}(s))$$

Grid example - Ex3 Tutorial 8

- Temporal Difference update for state $(1, 1)$, where $\gamma = 1$ and $\alpha = 0.1$.
- Temporal difference update rule:

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha(R(s) + \gamma U^{\pi}(s') - U^{\pi}(s))$$

- Let s' be $(1, 2)$.

Grid example - Ex3 Tutorial 8

- Temporal Difference update for state $(1,1)$, where $\gamma = 1$ and $\alpha = 0.1$.
- Temporal difference update rule:

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha(R(s) + \gamma U^{\pi}(s') - U^{\pi}(s))$$

- Let s' be $(1,2)$.
- If we use: $U((1,1)) = 0.71$, $U((1,2)) = 0.75$, $U((3,3)) = 0.912$, and $P((1,2)|(1,1), Up) = 0.75$, $P((1,1)|(1,1), Up) = 0.25$, $P((4,3)|(3,3), Right) = 0.6$, $P((3,2)|(3,3), Right) = 0.4$, then:

Grid example - Ex3 Tutorial 8

- Temporal Difference update for state $(1,1)$, where $\gamma = 1$ and $\alpha = 0.1$.
- Temporal difference update rule:

$$U^\pi(s) \leftarrow U^\pi(s) + \alpha(R(s) + \gamma U^\pi(s') - U^\pi(s))$$

- Let s' be $(1,2)$.
- If we use: $U((1,1)) = 0.71$, $U((1,2)) = 0.75$, $U((3,3)) = 0.912$, and $P((1,2)|(1,1), Up) = 0.75$, $P((1,1)|(1,1), Up) = 0.25$, $P((4,3)|(3,3), Right) = 0.6$, $P((3,2)|(3,3), Right) = 0.4$, then:

$$\begin{aligned} U^\pi((1,1)) &\leftarrow U^\pi((1,1)) + \alpha(R((1,1)) + \gamma U^\pi((1,2)) - U^\pi((1,1))) \\ &\leftarrow 0.71 + 0.1 \times (-0.04 + 1 \times 0.75 - 0.71) \\ &\leftarrow 0.71 + 0.1 \times 0 \\ &\leftarrow 0.71 \end{aligned}$$

- No change.

Grid example - Ex3 Tutorial 8

- Temporal Difference update for state $(3, 3)$, where $\gamma = 1$ and $\alpha = 0.1$.
- Temporal difference update rule:

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha(R(s) + \gamma U^{\pi}(s') - U^{\pi}(s))$$

Grid example - Ex3 Tutorial 8

- Temporal Difference update for state $(3, 3)$, where $\gamma = 1$ and $\alpha = 0.1$.
- Temporal difference update rule:

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha(R(s) + \gamma U^{\pi}(s') - U^{\pi}(s))$$

- Let s' be $(4, 3)$.

Grid example - Ex3 Tutorial 8

- Temporal Difference update for state $(3,3)$, where $\gamma = 1$ and $\alpha = 0.1$.
- Temporal difference update rule:

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha(R(s) + \gamma U^{\pi}(s') - U^{\pi}(s))$$

- Let s' be $(4,3)$.
- If we use: $U((1,1)) = 0.71$, $U((1,2)) = 0.75$, $U((3,3)) = 0.912$, and $P((1,2)|(1,1), Up) = 0.75$, $P((1,1)|(1,1), Up) = 0.25$, $P((4,3)|(3,3), Right) = 0.6$, $P((3,2)|(3,3), Right) = 0.4$, then:

Grid example - Ex3 Tutorial 8

- Temporal Difference update for state $(3,3)$, where $\gamma = 1$ and $\alpha = 0.1$.
- Temporal difference update rule:

$$U^\pi(s) \leftarrow U^\pi(s) + \alpha(R(s) + \gamma U^\pi(s') - U^\pi(s))$$

- Let s' be $(4,3)$.
- If we use: $U((1,1)) = 0.71$, $U((1,2)) = 0.75$, $U((3,3)) = 0.912$, and $P((1,2)|(1,1), Up) = 0.75$, $P((1,1)|(1,1), Up) = 0.25$, $P((4,3)|(3,3), Right) = 0.6$, $P((3,2)|(3,3), Right) = 0.4$, then:

$$\begin{aligned}
U^\pi((3,3)) &\leftarrow U^\pi((3,3)) + \alpha(R((3,3)) + \gamma U^\pi((4,3)) - U^\pi((3,3))) \\
&\leftarrow 0.912 + 0.1(-0.04 + 1 \times 1 - 0.912) \\
&\leftarrow 0.912 + 0.1 \times 0.048 \\
&\leftarrow 0.917
\end{aligned}$$

- The approximation we had for $U((3,3))$ improved.

Q-learning

function Q-LEARNING-AGENT(*percept*) **returns** an action
inputs: *percept*, a percept indicating the current state s' and reward signal r'
persistent: Q, a table of action values indexed by state and action, initially zero
N_{sa}, a table of frequencies for state–action pairs, initially zero
s, a, r, the previous state, action, and reward, initially null

if TERMINAL?(s) **then** $Q[s, None] \leftarrow r'$
if s is not null **then**
increment $N_{sa}[s, a]$
$Q[s,a] \leftarrow Q[s,a] + \alpha(N_{sa}[s,a])(r + \gamma \max_{a'} Q[s',a'] - Q[s,a])$
$s, a, r \leftarrow s', \text{argmax}_{a'} f(Q[s',a'], N_{sa}[s',a']), r'$
return a

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

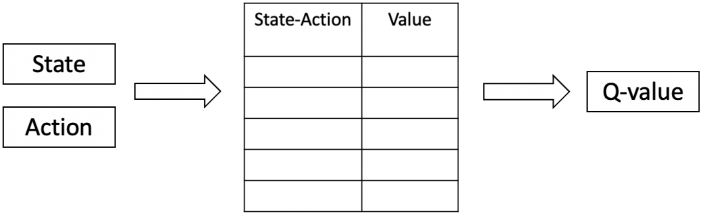
Deep Q-Network (DQN)

- Q: Could you explain a little bit about Deep Q-Learning?

DQN

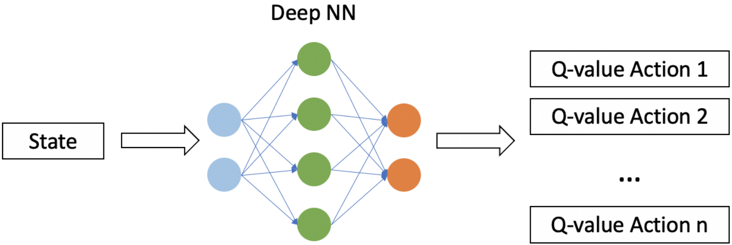
- Q-learning: lookup table.
- Table mapping each state-action pair to a Q-value.

Q table



DQN

- Instead of table, we have a neural network that maps states to action-Q-value pairs.



DQN

- *Playing Atari with Deep Reinforcement Learning, DeepMind, 2013*
- Application: Atari 2600 games
- CNNs trained with Q-learning
- CNNs learn control policies from raw video data in complex RL environments
- Input: state representation
- Output: separate output unit for each possible action, predicted Q-values of the individual action for the input state
- Experience replay: store the agent's experiences at each time-step over episodes
- After experience replay, the agent selects and executes an action using ϵ-greedy policy

KING'S College LONDON