Mastering Imperfect State Games through Deep Recurrent Reinforcement Learning

DAI, Zhiwen
SEEM, CUHK
18 April 2019

Intro: Problem Setting

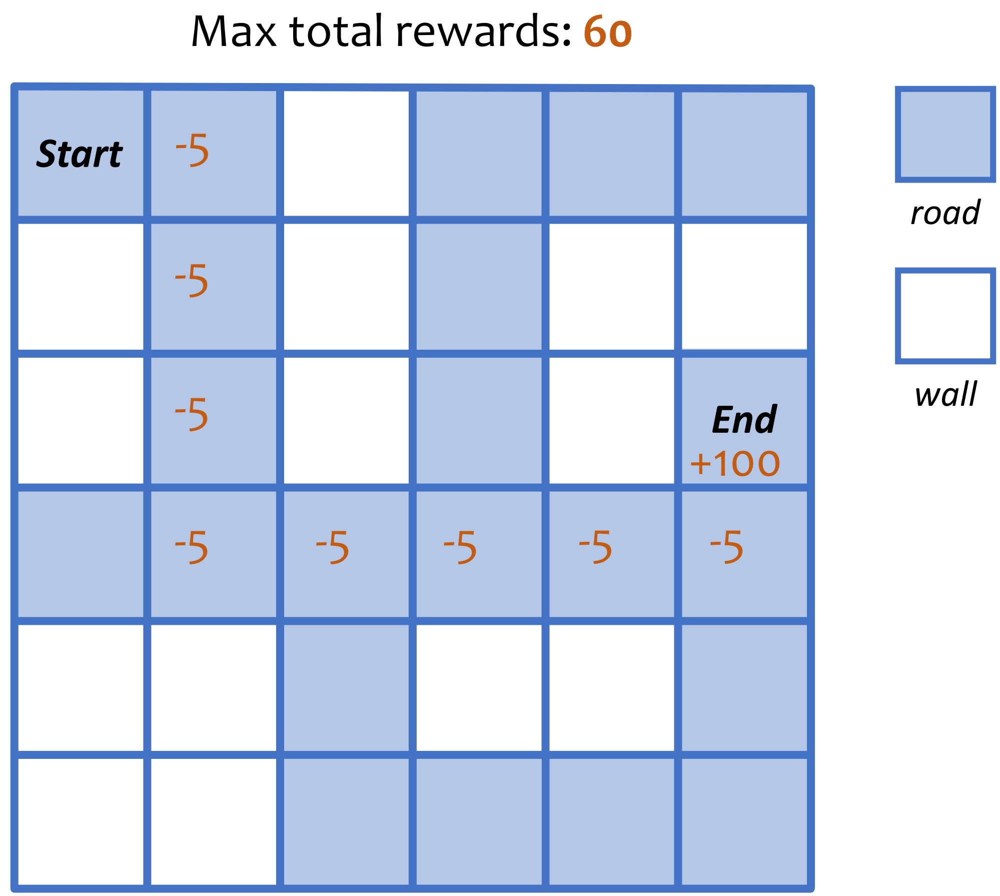
- We often face **imperfect state** control problems.
- MDP → **Partial Observable MDP** (POMDP)
- In control, we can use the **historical information** I_k to predict the current state S_k.
- However, sometimes we even don't know **what S_k is**, so we can't do the prediction.
- We may even don't know the **transition probabilities** of the POMDP.
- In a word, we can only know what is **observable** (observations, actions, rewards).

Intro: Reinforcement Learning

- **RL** can deal with
  - MDPs **not knowing the transition probabilities** (since it's model-free).
  - **huge MDP models** (video games, Go...) using deep learning (**DQN**, A2C, PPO…)
- How about **POMDP**?
- I will introduce some new algorithms in RL to deal with POMDPs.
  - **Deep Recurrent Q-Network** (literature review)
  - **Deep Recurrent Q-Network with Actions** (I proposed)
- I will introduce them using a game **MazeWorld**.

MazeWorld: Level 1

- Start from (0, 0).
- End at (2, 5).
- State: (row, column)
- Reward: **-5** each step before End; **100** at End.
- Goal: maximize total rewards.
- It's a **perfect state MDP**.

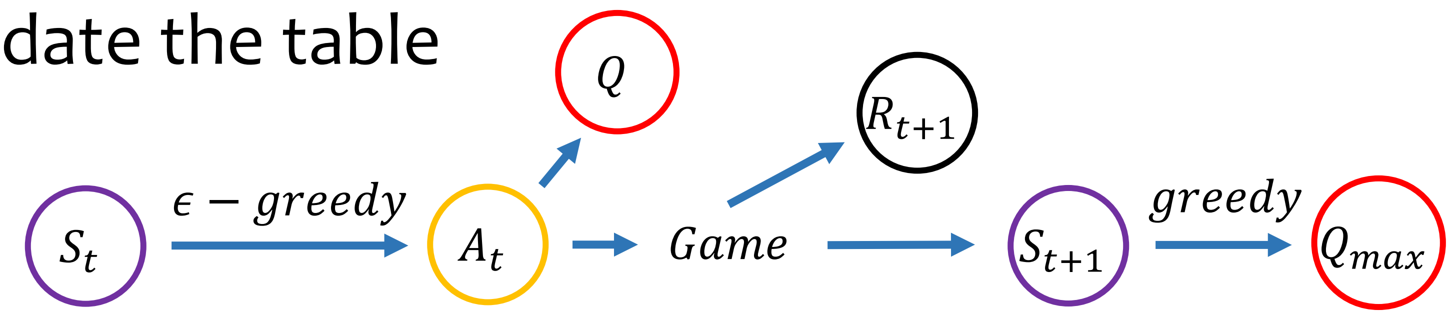


Algorithm 1: Q-learning

- Q-table

	UP	DOWN	LEFT	RIGHT
(0,0)	-5	-3	-10	10
(0,1)	-5	5	-3	-4
…	…	…	…	…
(5,5)	…	…	…	…

- Update the table

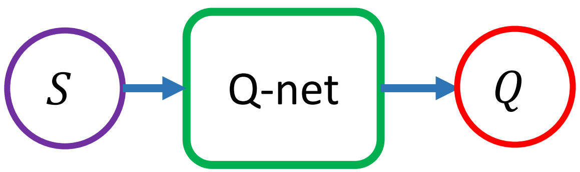


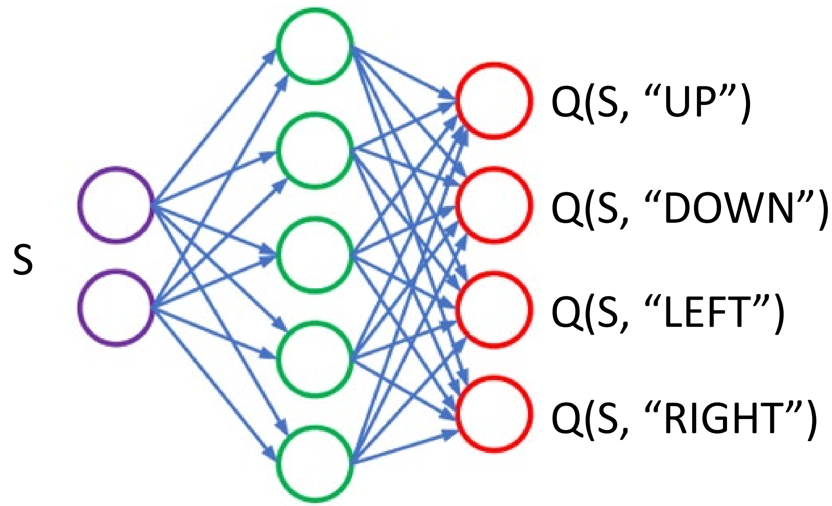
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \Big].$$

Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

Algorithm 2: Deep Q-Network

- What if the Q-table is too large?
- Use a **neural network** to substitute the Q-table.
- Q-network





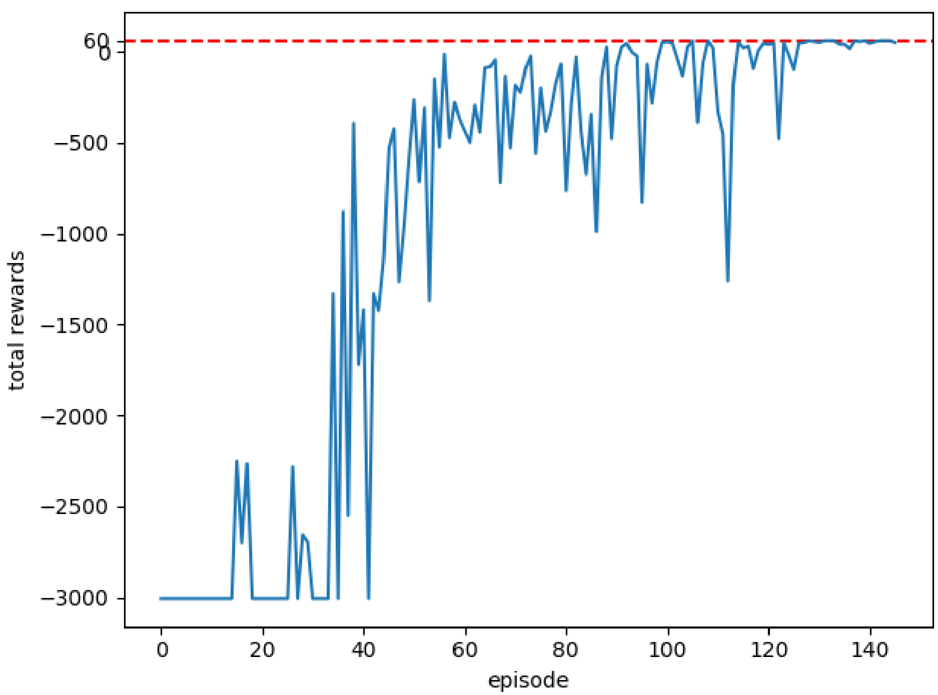
- Train the Q-network
  - Build a "target network"
  - Sample from memory
  - Do the updates

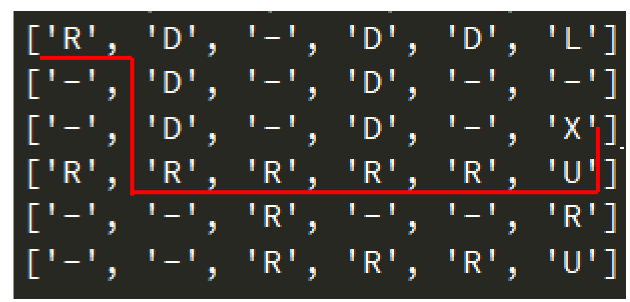
Details are in this famous paper by DeepMind

Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *Nature* 518.7540 (2015): 529.

Results of Level 1

- DQN
  - input: 2 →hidden1: 20 → hidden2: 50 → output: 4.





```
['R', 'D', '-', 'D', 'D', 'L']
['-', 'D', '-', 'D', '-', '-']
['-', 'D', '-', 'D', '-', 'X']
['R', 'R', 'R', 'R', 'R', 'U']
['-', '-', 'R', '-', '-', 'R']
['-', '-', 'R', 'R', 'R', 'U']
```

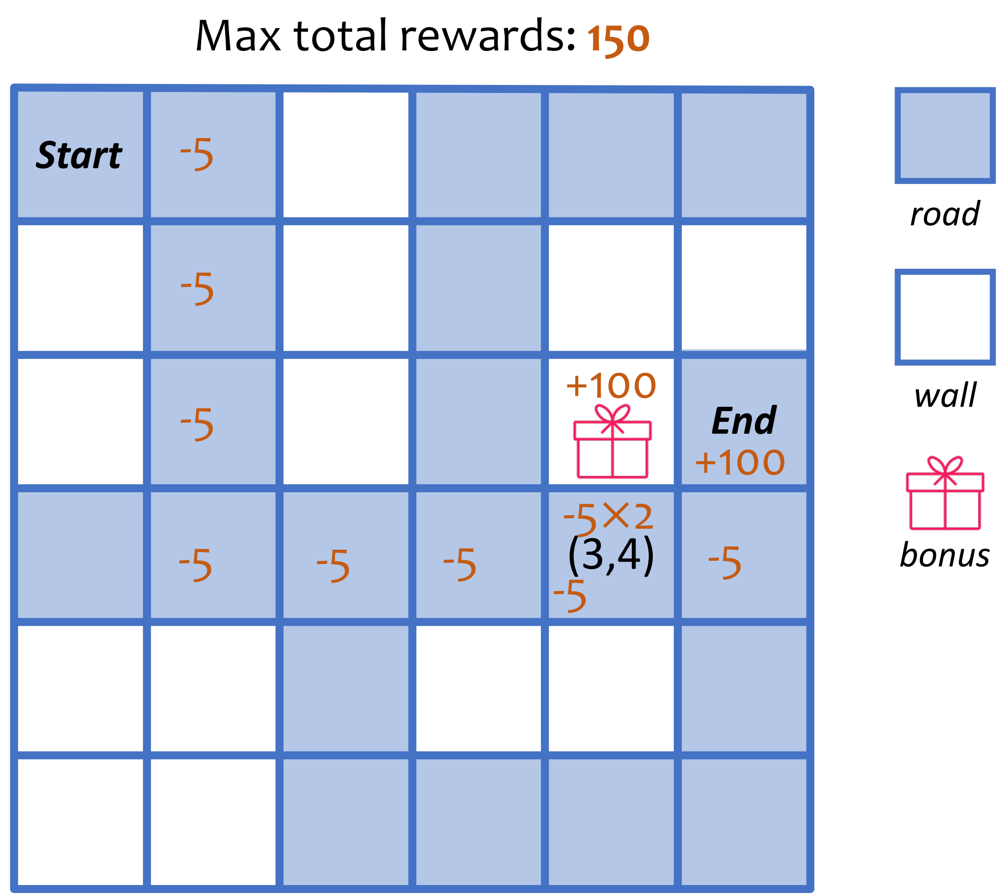
Strategy of DQN

MazeWorld: Level 2

- **Bonus 100** at (3, 4), go "UP" twice to get it.
- Goal: maximize total rewards.
- It's an **POMDP**.
- True state is:

(row, column, #doing "UP" at (3, 4))

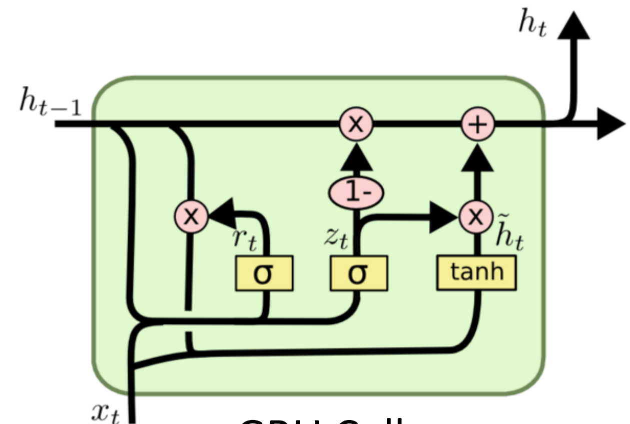
- Observation: (r, c)
- DQN? Not working well !



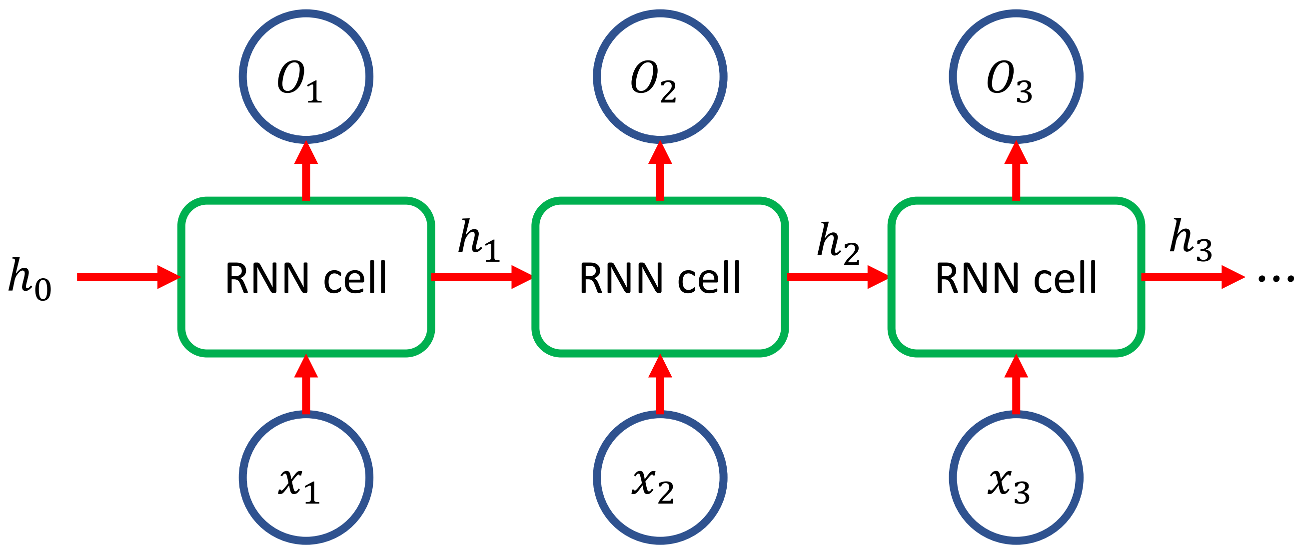
Algorithm 3: Deep Recurrent Q-Network

My notes of RNN: https://github.com/Daizhiwen/Intro-to-Recurrent-Neural-Networks

- Recurrent Neural Networks
  - RNN cells: Basic RNN, GRU, LSTM…
  - Gated Recurrent Unit can learn to memory and forget things (like LSTM, but simpler).



GRU Cell

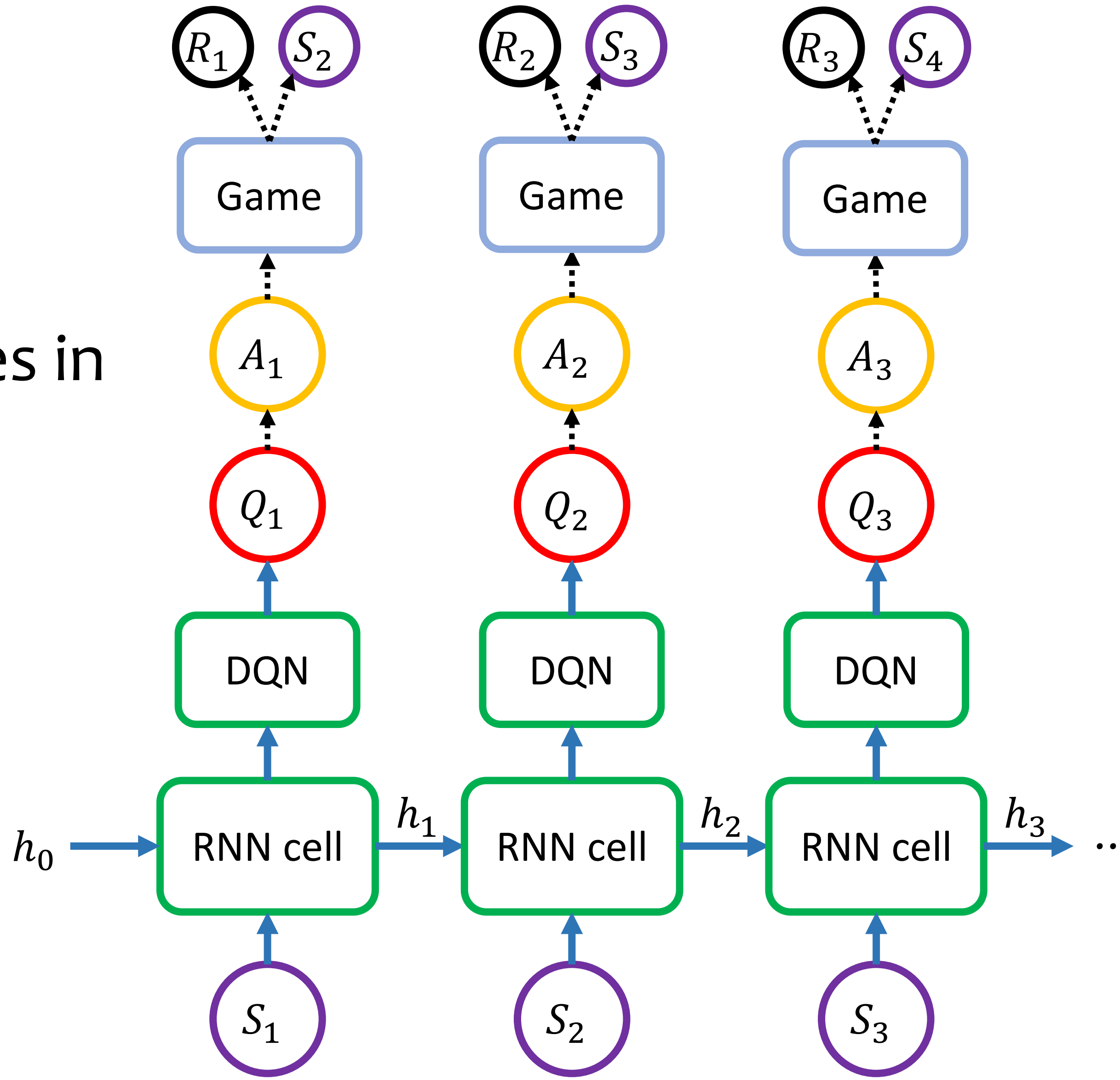


- h_1 depends on x_1
- h_2 depends on x_1 and x_2
- …

Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." *arXiv preprint arXiv:1406.1078* (2014).

Algorithm 3: Deep Recurrent Q-Network, cont'd

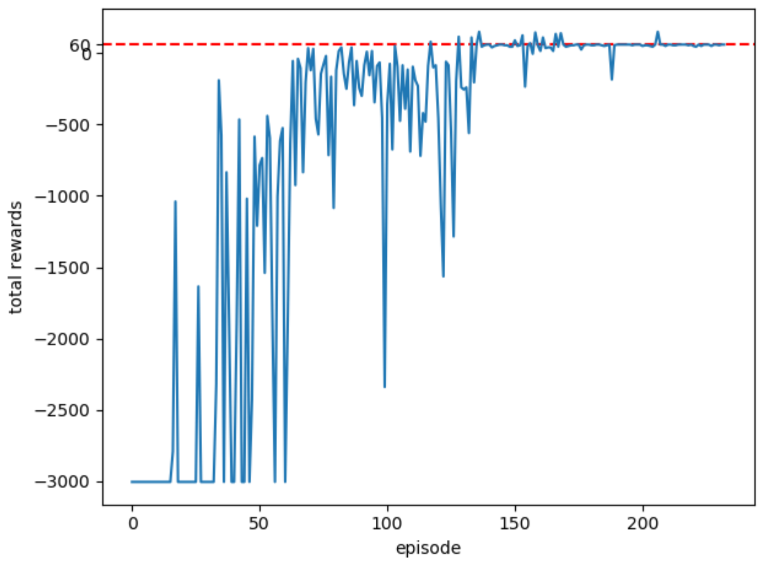
- DRQN
- $h_0 = \vec{0}$
- Some changes in the training from DQN.



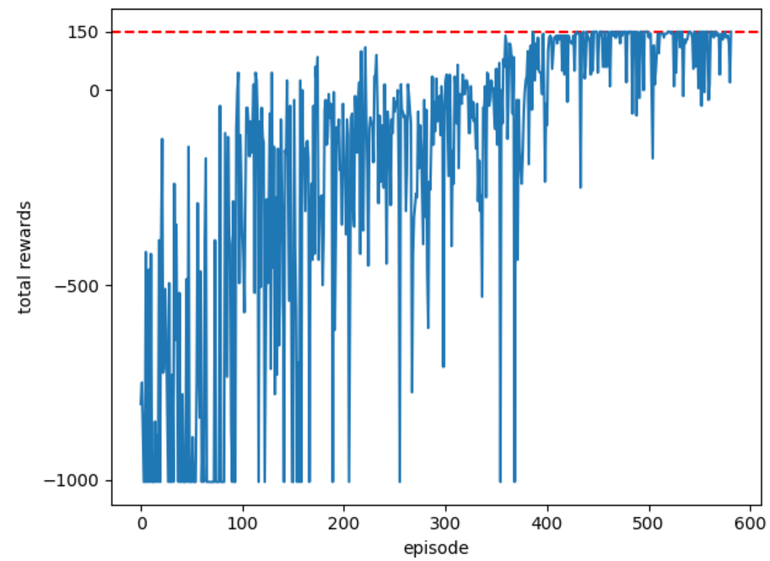
Hausknecht, Matthew, and Peter Stone. "Deep recurrent q-learning for partially observable mdps." *2015 AAAI Fall Symposium Series*. 2015.

Results of Level 2

- DQN
  - input: 2 →hidden1: 20 → hidden2: 50 → output: 4.
- DRQN
  - GRU size: 10, DQN: 10 → 50 → 50 → 4.



DQN: Converges, but not optimal



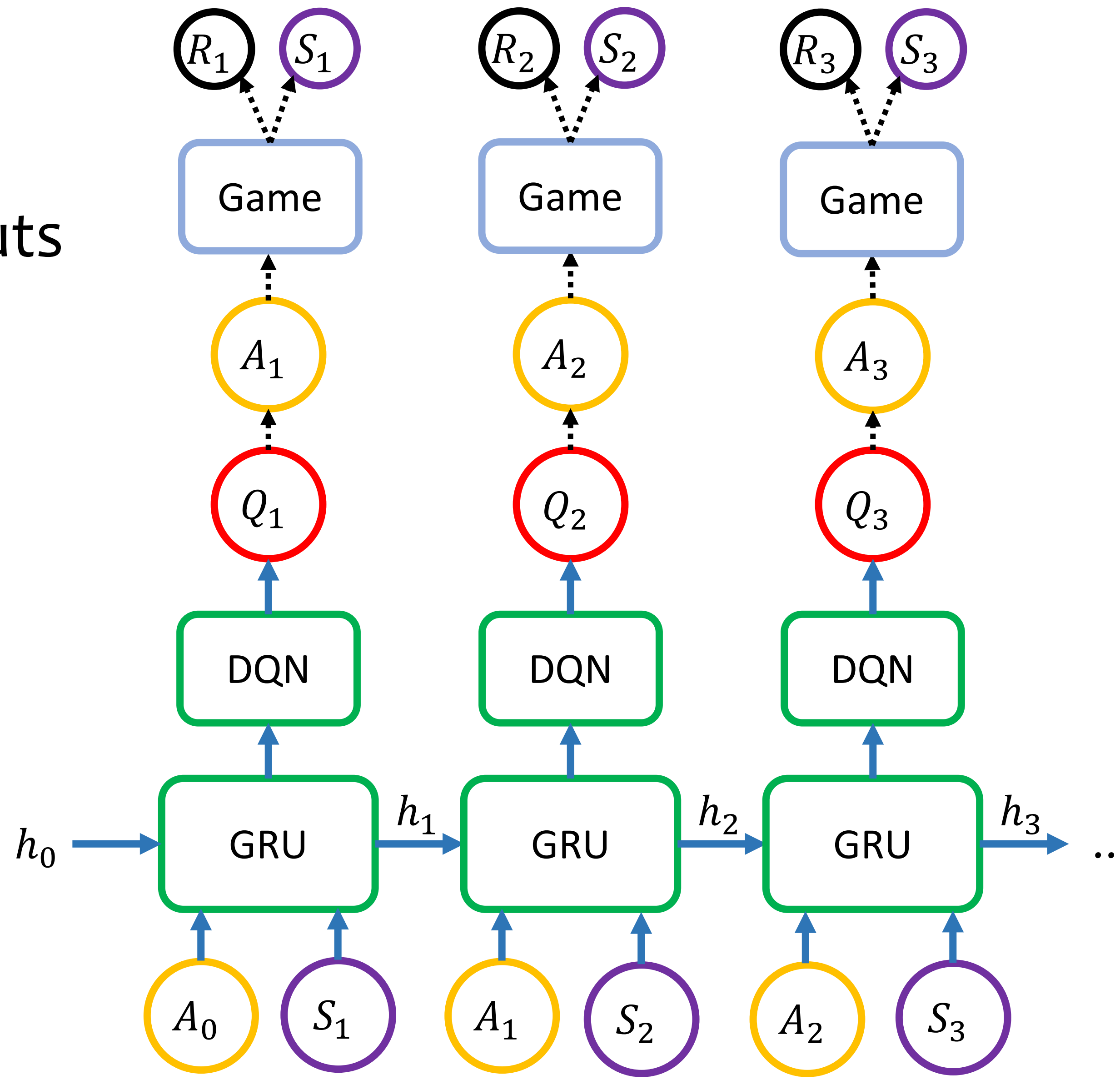
DRQN: Converges to optimum

MazeWorld: Level 3

- Same maze as in Level 2.
- But we can **only** observe the **row information**!
- It's reasonable that knowing only the history of row information is not very helpful.
- What if we know the **history of actions**?
- Knowing **row** and **action,** we may guess the **column** better!

Algorithm 4: DRQN + Actions

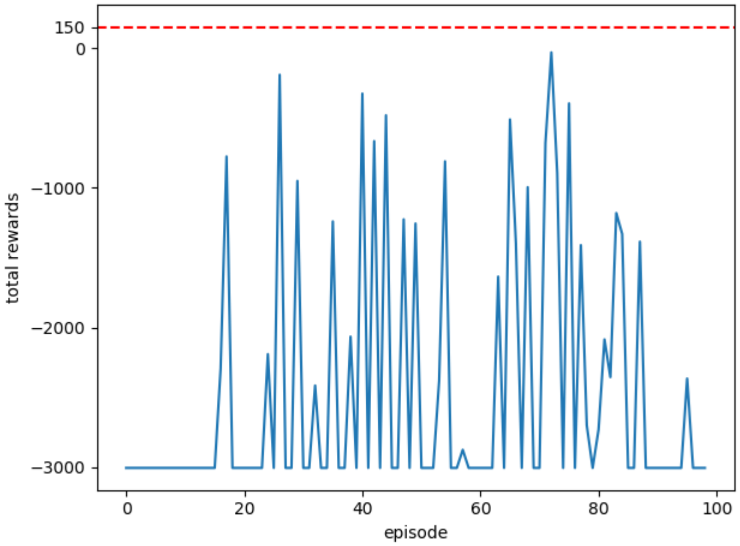
- Add actions into the inputs of GRUs.
- $A_0 = \vec{0}$
- $h_0 = \vec{0}$



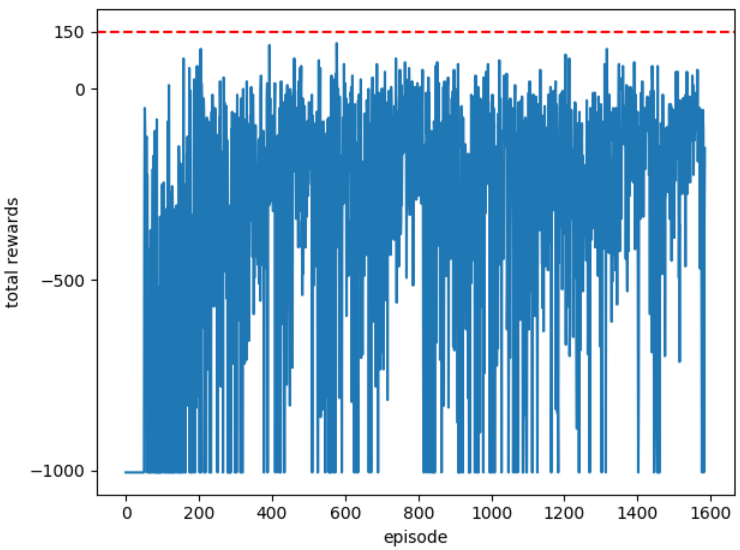
Results of Level 3

- DQN
  - input: 2 →hidden1: 20 → hidden2: 50 → output: 4.
- DRQN
  - GRU size: 10, 1 layer. DQN: 10 → 50 → 50 → 4.
- DRQN+A
  - GRU size: 15, 2 layers. DQN: 15 → 50 → 50 → 4.

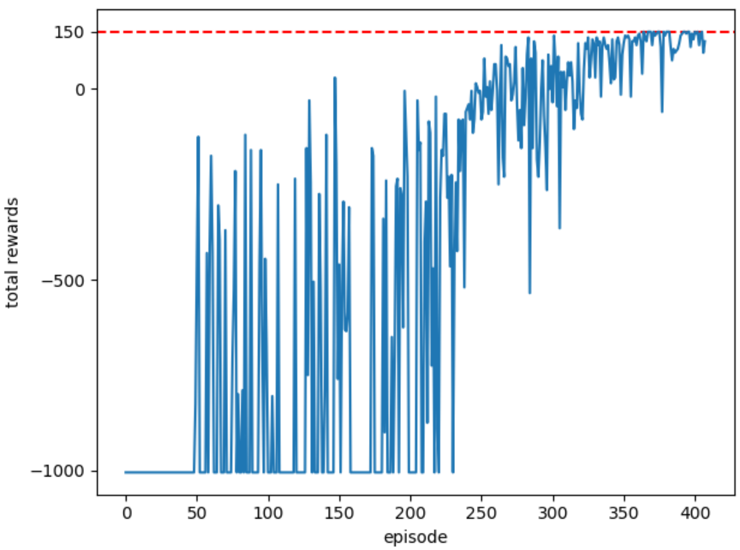
Results of Level 3, cont'd



DQN
Not converges



DRQN: Not converges



DRQN+A: Converges to optimum

Summary

- Game: MazeWorld, level 1: MDP, level 2&3: POMDP
- Algorithms: DQN, DRQN, DRQN+A
- Performance:

  but converges to a suboptimum

DQN:	level 1 √	level 2 ×	level 3 ×
DRQN:	level 1 √	level 2 √	level 3 ×
DRQN+A:	level 1 √	level 2 √	level 3 √

- View my codes (PyTorch) on GitHub: https://github.com/dull-bird/drqn_mazeworld
- DQN & DRQN codes references: https://github.com/metalbubble/DeepRL-Tutorials

Thank You!