Deep RL Learning applied to Pac-Man

Lusine Kamikyan and Emma Reid

Purdue University

December 10, 2018

Outline

- Reinforcement Learning
- Q-learning algorithm
- Deep Q-learning
- Double DQN
- Dueling DQN
- Environment
- Network Architecture
- Training
- Results

Reinforcement Learning

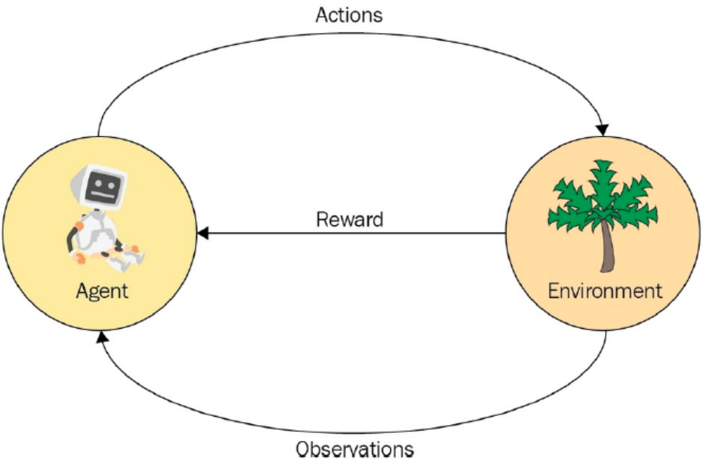


Figure: Elements or RL

Q Learning

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	327	0	0	0	0	0	0
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	499	0	0	0	0	0	0

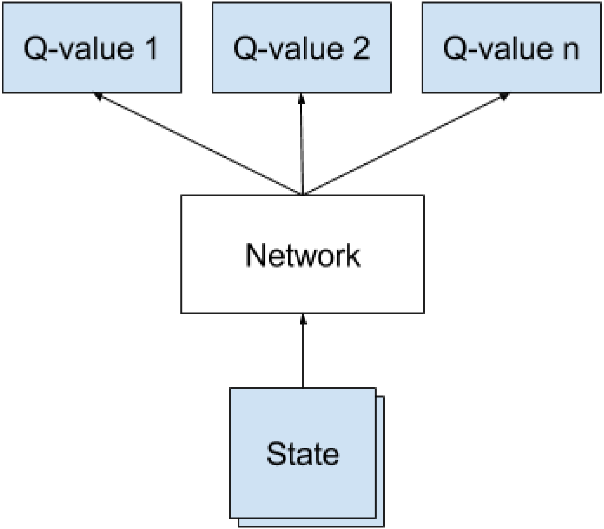
Training

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	328	-2.30108105	-1.97092096	-2.30357004	-2.20591839	-10.3607344	-8.5583017
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	499	9.96984239	4.02706992	12.96022777	29	3.32877873	3.38230603

$$Q^{new}(s_t,a_t) \leftarrow (1-\alpha)\cdot \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\bigg(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}} \bigg)}^{\text{learned value}}$$

Figure: Q Learning Table

Deep Q Learning



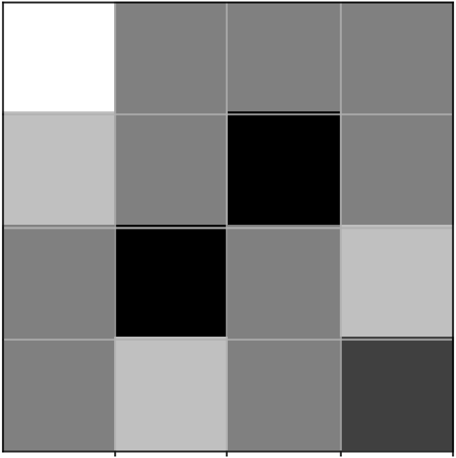
Deep Q Learning

- ϵ**-greedy:** helps to explore the environment at the beginning and to stick to a good policy at the end
- **Replay buffer:** storing the previous experiences in a large buffer, sampling training data from it instead of using our latest experience
- **Target Network:** keep a copy of the network and use it for the value of $Q(s', a')$ in the Bellman equation.

Deep Q Learning Algorithm

1. Initialize replay memory D to capacity N
2. Initialize action-value function Q with random weight θ
3. Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
4. **For** episode 1:M **do**
5. Initialize sequence $\{s_1\}$
6. **For** t = 1:T **do**
7. With probability ϵ select a random action a_t, otherwise $a_t = argmax_a Q(s_t, a; \theta)$
8. Execute action a_t and observe reward r_t and the next state s_{t+1}
9. Store transition (s_t, a_t, r_t, s_{t+1}) in D
10. Sample a random minibatch of transitions from D
11. For every transition in the buffer, calculate target $y = r$ if the episode has ended at this step or $y = r + \gamma max_{a \in A} \hat{Q}(s_{t+1}, a; \theta^{-})$ otherwise
12. Calculate loss: $L = (Q(s_t, a_t; \theta) - y)^2$
13. Update $Q(s, a)$ using SGD algorithm by minimizing the loss in respect to model parameters θ
14. Every C steps reset $\hat{Q} = Q$

Environment of Pac-Man



Network Architecture

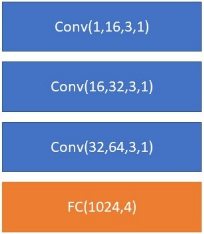


Figure: DQN and Double DQN

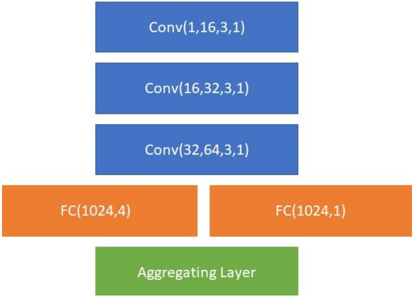
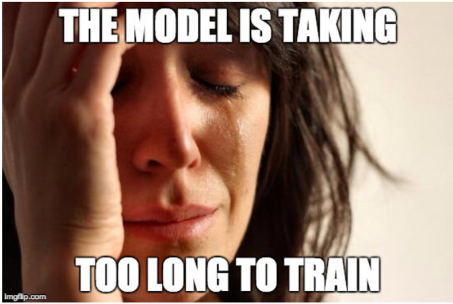


Figure: Dueling DQN

DQN: $y = r + \gamma max_{a \in A} \hat{Q}(s_{t+1}, a; \theta^{-})$
Double DQN: $Q(s_t, a_t) = r_t + \gamma \hat{Q}(s_{t+1}, argmax_a Q(s_{t+1}, a))$

Results



DQN Loss, Rewards, Win number for 4 by 4 grid, with 3 goals, 1 ghost

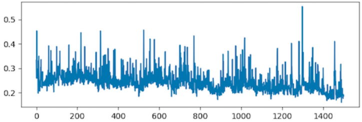


Figure: DQN loss for 4 by 4 after 2500 episodes

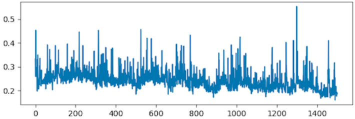


Figure: DQN loss for 4 by 4 after another 1500 episodes

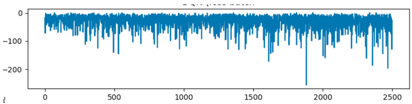


Figure: DQN Rewards

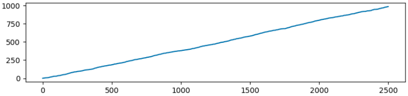


Figure: DQN win number

Double DQN and Dueling Double DQN Losses for 4 by 4 grid, 3 goals, 1 ghost

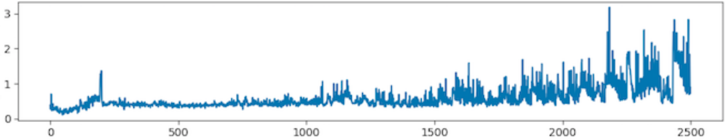


Figure: Double DQN Loss

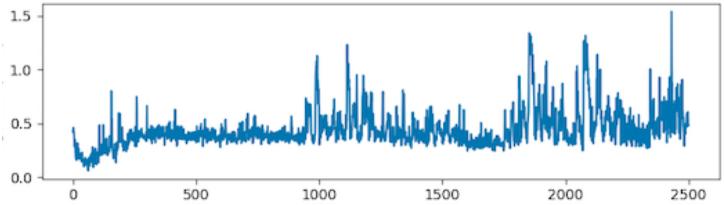


Figure: Dueling Double DQN Loss

Reasons why the agent didn't learn

- Need to run for much longer time.
- Need more positive experiences in order to learn better - having small batch size (due to computing power limitation) didn't help with the issue
- Need to decrease the epsilon only when we win and decrease slowly enough in order for the agents to learn.

Conclusions and Future Work

- **Conclusion**
  - DQN takes a **very** long time to train Pac-Man
  - Needs lots of data and the right data
  - DQN seemed to be better from the initial runs, however, we expect the Double and Dueling DQN to become better as we run longer
- **Future Work**
  - Try Prioritized Experience Replay
  - Train with more goals and more ghosts
  - Train the ghosts move randomly
  - Train the ghost try to catch the Pac-Man
  - Have the same NN control both the agent and the ghosts
  - Have two different NNs control the ghosts and the agent

References

- $cs229.stanford.edu/proj2017/final-reports/5241109.pdf$
- $towardsdatascience.com/advanced-dqns-playing-pac-man-with-deep-reinforcement-learning-3ffbd99e0814$
- $pytorch.org/tutorials/intermediate/reinforcement\_q\_learning$
- $github.com/udacity/deep-reinforcement-learning$
- $gist.github.com/simoninithomas/7611db5d8a6f3edde269e18b97fa4d0c$
- $samyzaf.com/ML/tdf/tdf.html$
- $morvanzhou.github.io/tutorials/$
- $https://medium.freecodecamp.org/improvements-in-deep-q-learning-dueling-double-dqn-prioritized-experience-replay-and-fixed-58b130cc5682$