AIGSA Reading Group

Atari Environments + DQN

Recap

- MDP
- Bellman Backup
- Value Iteration
- Policy Iteration

Recap

- MDP
- Bellman Backup
- Value Iteration
- Policy Iteration

Today

- DQN + Atari

Optimal Value Function

- An optimal value function is the maximum achievable value

$$Q^*(s,a) = \max_\pi Q^\pi(s,a) = Q^{\pi^*}(s,a)$$

- Once we have Q*, the agent can act optimally

$$\pi^*(s) = \operatorname*{argmax}_a Q^*(s,a)$$

- Formally given by Bellman equation

$$Q^*(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^*(s',a') \mid s,a\right]$$

Q-Learning

- Model-free, off-policy learning
- The algorithm maintains a table/function of Q-value such that each entry, $Q(s, a) \rightarrow$ Expected cumulative reward for taking action *a* in *s*, and following the optimal policy thereafter
- Q-value update rule:
  $Q(s, a) \leftarrow Q(s, a) + \alpha\,(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$

Q-Learning

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$
    until S is terminal

Q-Learning with Function Approximation

Setting: Large or continuous state spaces

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

Minimize MSE loss by stochastic gradient descent

$$J(\mathbf{w}) = E_{\pi}[(Q_{\pi}(S,A) - Q(S,A,\mathbf{w}))^2]$$

$$J(\mathbf{w}) = (r + \gamma \max_{a'} Q(s', a', \mathbf{w}) - Q(s, a, \mathbf{w}))^2$$

Q-Learning with Function Approximation

Minimize MSE loss by stochastic gradient descent

$$J(\mathbf{w}) = (r + \gamma \max_{a'} Q(s', a', \mathbf{w}) - Q(s, a, \mathbf{w}))^2$$

Problem – Does not converge when,

- Samples are correlated
- Target is non-stationary

Playing Atari with Deep Reinforcement Learning, Mnih et al., 2013

DQN

- To remove correlations, use replay buffer

s_1, a_1, r_2, s_2
s_2, a_2, r_3, s_3
s_3, a_3, r_4, s_4
$\ldots$
$s_t, a_t, r_{t+1}, s_{t+1}$

← a fixed size buffer

Sample experiences and apply update

$$J(\mathbf{w}) = (r + \gamma \max_{a'} Q(s', a', \mathbf{w}^-) - Q(s, a, \mathbf{w}))^2$$

- To deal with non-stationarity,
  - DQN uses maintains a distinct *target network*
  - Target network weights $\mathbf{w}^-$, are held fixed
  - $\mathbf{w}^-$ is updated after certain training steps to match $\mathbf{w}$ of the main Q-network

DQN

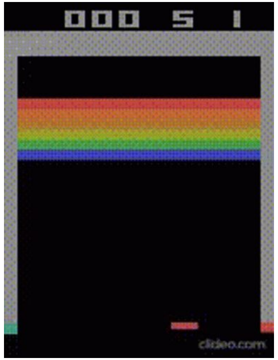
Overall idea:

- ➔ Take action a_t with ε-probability
- ➔ Store transitions $(s_t, a_t, r_{t+1}, s_{t+1})$ in replay buffer D
- ➔ Sample random mini-batch of transitions (s,a,r,s’) from D
- ➔ Compute Q-learning targets w.r.t. old, fixed $\mathbf{w}^-$
- ➔ Optimize MSE between Q-network & Q-learning targets

$$J(\mathbf{w}_i) = \mathbb{E}_{s,a,r,s' \sim D}\Big[\Big(\underbrace{\big(r + \gamma \max_{a'} Q(s', a'; \mathbf{w}^-)\big)}_{\text{Q-learning target}} - \underbrace{Q(s, a; \mathbf{w}_i)}_{\text{Q-network}}\Big)^2\Big]$$

- ➔ Update **w** using stochastic gradient descent

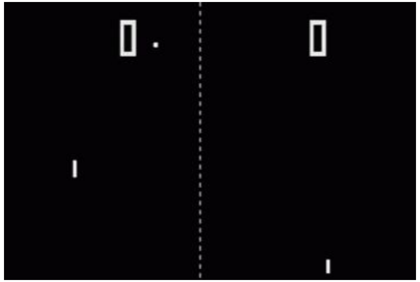
Atari


000 5 1
clideo.com


PLAYER 1
21340
CHANGE TO:
LEVEL: 2
ROUND: 3


SCORE<1> HI-SCORE SCORE<2>
0000 0000
3
CREDIT 00


6
0:18.63
ACTIVISION 1


0
0

Atari

You can use it to test your algorithms…. Today we will test DQN on the atari env

References

1. Playing Atari with Deep Reinforcement Learning, Mnih et al., 2013
2. The Arcade Learning Environment (ALE), Bellemare et al., 2012
3. Q-learning, Christopher JCH Watkins and Peter Dayan, 1992
4. CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms, Huang et al., 2022