TTIC 31230, Fundamentals of Deep Learning

David McAllester, Autumn 2020

Reinforcement Learning

Q-Learning

Review

- **A Policy** π is a stochastic way of selection an action at a state.

- **Imitation Learning** (cross entropy imitation of action given state).

- Imitation Learning is **off-policy**.

- The **value function** $V^{\pi}(s)$.

- **Value Iteration** $V_{i+1}(s) = \text{argmax}_a \; R(s,a)+\gamma E_{s'} \; \gamma V_i(s')$

The Q Function

For discounted reward:

$$Q^\pi(s,a) = E_\pi \sum_t \gamma^t r_t \mid \pi,\ s_0 = s,\ a_0 = a$$

$$Q^*(s,a) = \sup_\pi Q^\pi(s,a)$$

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

$$Q^*(s,a) = R(s,a) + \gamma E_{s' \sim P_T(\cdot|s,a)} \max_{a'} Q^*(s',a')$$

Q Function Iteration

It is possible to define Q-iteration by analogy with value iteration, but this is generally not discussed.

Value iteration is typically done for finite state spaces. Let S be the number of states and A be the number of actions.

One update of a Q table takes $O(S^2A^2)$ time while one update of value iteration is $O(S^2A)$.

Q-Learning

When learning by updating the Q function we typically assume a parameterized Q function $Q_\Phi(s,a)$.

Bellman Error:

$$\mathrm{Bell}_\Phi(s,a) \doteq \left(Q_\Phi(s,a) - \left(R(s,a) + \gamma\ E_{s' \sim P_T(s'|s,a)}\ \max_{a'}\ Q_\Phi(s',a')\right)\right)^2$$

Theorem: If $\mathrm{Bell}_\Phi(s,a) = 0$ for all (s,a) then the induced policy is optimal.

Algorithm: Generate pairs (s,a) from the policy $\mathrm{argmax}_a\ Q_\Phi(s_t,a)$ and repeat

$$\Phi \mathrel{-}= \eta \nabla_\Phi\ \mathrm{Bell}_\Phi(s,a)$$

Issues with Q-Learning

Problem 1: Nearby states in the same run are highly correlated. This increases the variance of the cumulative gradient updates.

Problem 2: SGD on Bellman error tends to be unstable. Failure of Q_Φ to model unused actions leads to policy change (exploration). But this causes Q_Φ to stop modeling the previous actions which causes the policy to change back ...

To address these problems we can use a **replay buffer**.

Using a Replay Buffer

We use a replay buffer of tuples (s_t, a_t, r_t, s_{t+1}).

Repeat:

1. Run the policy $\mathrm{argmax}_a\, Q_\Phi(s,a)$ to add tuples to the replay buffer. Remove oldest tuples to maintain a maximum buffer size.
2. $\Psi = \Phi$
3. for N times select a random element of the replay buffer and do

$$\Phi \mathrel{-}= \eta \nabla_\Phi \left(Q_\Phi(s_t, a_t) - (r_t + \gamma \max_a Q_\Psi(s_{t+1}, a)\right)^2$$

Replay is Off-Policy

Note that the replay buffer is from a **mixture of policies** and is **off-policy** for $\mathrm{argmax}_a\ Q_\Phi(s,a)$. This seems to be important for stability.

This seems related to the issue of stochastic vs. deterministic policies. More on this later.

Multi-Step Q-learning

$$\Phi \mathrel{-}= \sum_t \nabla_\Phi \left(Q_\Phi(s_t, a_t) - \sum_{\delta=0}^{D} \gamma^\delta r_{(t+\delta)} \right)^2$$

Asynchronous Q-Learning (Simplified)

No replay buffer. Many asynchronous threads each repeating:

$\tilde{\Phi} = \Phi$ (retrieve Φ)

using policy $\mathrm{argmax}_a\, Q_{\tilde{\Phi}}(s,a)$ compute

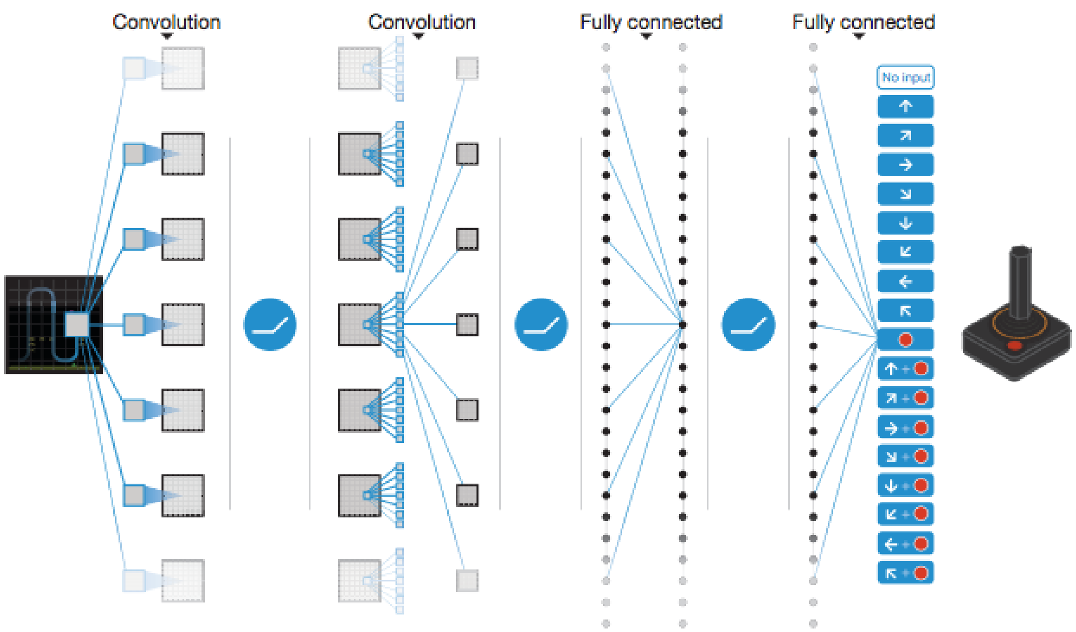
$$s_t, a_t, r_t, \ldots, s_{t+K}, a_{t+K}, r_{t+K}$$

Φ -= $\eta \sum_{i=t}^{t+K-D} \nabla_{\tilde{\Phi}} \left(Q_{\tilde{\Phi}}(s_i,a_i) - \sum_{\delta=0}^{D} \gamma^\delta r_{i+\delta}\right)^2$ (update Φ)

Human-level control through deep RL (DQN)

Mnih et al., Nature, 2015. (Deep Mind)

We consider a CNN $Q_\Phi(s, a)$.



Watch The Video

https://www.youtube.com/watch?v=V1eYniJ0Rnk

END