Non-linear Value Function Approximation: Deep Q-Network

Alina Vereshchaka

CSE4/510 Reinforcement Learning
Fall 2019

avereshc@buffalo.edu

October 1, 2019

Overview

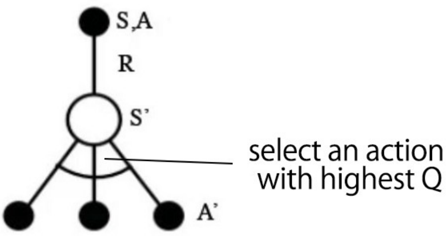
Table of Contents

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

- Learn to master 49 different Atari games from screens
- Excel human experts in 29 games
- Uses Deep Q-network receiving only the pixels and the game score as inputs

Recap: Q-Learning Algorithm

- Q-learning learns the action-value function $Q(s, a)$: how good to take an action at a particular state.
- From the memory table, we determine the next action a' to take which has the maximum $Q(s', a')$.



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'

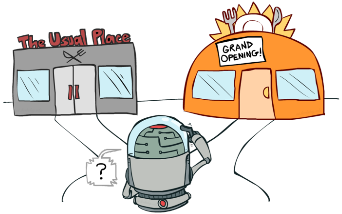
  $$Q(S, A) \leftarrow Q(S, A) + \alpha \Big[\underbrace{\overbrace{\boxed{\underset{\text{Immediate Reward}}{R} + \gamma \max_a Q(S', a)}}^{\text{Target}} - \overbrace{\boxed{Q(S, A)}}^{\text{Prediction}}}_{\text{loss}} \Big]$$

  $S \leftarrow S'$

until S is terminal

Recap: Exploration vs Exploitation

- Exploration
  - Discover better action selections
  - Improve the knowledge about the environment
  - ϵ-greedy exploration – with probability ϵ choose a random action, otherwise go with the "greedy" action with the highest Q-value.
- Exploitation
  - Maximize the reward based on what agent already knows



Exploration vs Exploitation

$\epsilon -$ *greedy* algorithm:

- With probability ϵ choose a random action a
- With probability $1 - \epsilon$ choose "greedy" action a with the highest Q-value.

Deep Q-Network (DQN): AI = RL + DL

- Reinforcement Learning(RL) defines the **objective**
- Deep Learning(DL) gives the **mechanism**

RL + DL = General Intelligence

Deep Q-Network (DQN)

- Represent value function by deep Q-network with weights w

$$Q(s, a, w) \approx Q^{\pi}(s, a)$$

Deep Q-Network (DQN)

- Represent value function by deep Q-network with weights w

$$Q(s, a, w) \approx Q^{\pi}(s, a)$$

- Define objective function

$$\mathcal{L}(w) = \mathbb{E}\left[\left(\underbrace{r + \gamma \max_{a'} Q(s', a', w)}_{\text{target}} - Q(s, a, w)\right)^2\right]$$

Deep Q-Network (DQN)

- Represent value function by deep Q-network with weights w

$$Q(s, a, w) \approx Q^{\pi}(s, a)$$

- Define objective function

$$\mathcal{L}(w) = \mathbb{E}\left[\left(\underbrace{r + \gamma \max_{a'} Q(s', a', w)}_{\text{target}} - Q(s, a, w)\right)^2\right]$$

- Leading to the following Q-leaning gradient

$$\frac{\partial \mathcal{L}(w)}{\partial w} = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a', w) - Q(s, a, w)\right) \frac{\partial Q(s, a, w)}{\partial w}\right]$$

Deep Q-Network (DQN)

- Represent value function by deep Q-network with weights w

$$Q(s, a, w) \approx Q^{\pi}(s, a)$$

- Define objective function

$$\mathcal{L}(w) = \mathbb{E}\left[\left(\underbrace{r + \gamma \max_{a'} Q(s', a', w)}_{\text{target}} - Q(s, a, w)\right)^2\right]$$

- Leading to the following Q-leaning gradient

$$\frac{\partial \mathcal{L}(w)}{\partial w} = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a', w) - Q(s, a, w)\right) \frac{\partial Q(s, a, w)}{\partial w}\right]$$

- Optimize objective end-to-end by SGD, using $\frac{\partial L(w)}{\partial w}$

Stability issues with Deep RL

Naive Q-learning oscillates or diverge with neural nets

1. Data is sequential
   - Successive samples are correlated, non-iid
2. Policy changes rapidly with slight changes to Q-values
   - Policy can oscillate
   - Disctribution of data can swing from one extreme to another
3. Scale of reqards and Q-values is unknown
   - Naive Q-learning gradients can be large unstable when backpropagated

Recap: Independent and Identically Distributed (iid) data

In supervised learning, we want the input to be i.i.d. (independent and identically distributed):

- Samples are randomized among batches and therefore each batch has the same (or similar) data distribution
- Samples are independent of each other in the same batch. If not, the model may be overfitted for some class (or groups) of samples at different time and the solution will not be generalized.

Deep Q-Networks

DQN provides a stable solution to deep value-based RL

1. Use experience replay
   - Break correlations in data, bring us back to iid setting
   - Learn from all past policies
2. Freeze target Q-network
   - Avoid oscillations
   - Break correlations between Q-network and target
3. Clip rewards or normalize network adaptive to sensible range
   - Robust gradients

Stable Deep RL (1): Experience Replay

Problem: approximation of Q-values using non-linear functions is not stable

Solution:

- Take action a_t according to ϵ-greedy policy

Stable Deep RL (1): Experience Replay

Problem: approximation of Q-values using non-linear functions is not stable

Solution:

- Take action a_t according to ϵ-greedy policy
- Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in a replay memory D

Stable Deep RL (1): Experience Replay

Problem: approximation of Q-values using non-linear functions is not stable

Solution:

- Take action a_t according to ϵ-greedy policy
- Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in a replay memory D
- Sample random mini-batch of transitions $(s_t, a_t, r_{t+1}, s_{t+1})$ from D

Stable Deep RL (1): Experience Replay

Problem: approximation of Q-values using non-linear functions is not stable

Solution:

- Take action a_t according to ϵ-greedy policy
- Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in a replay memory D
- Sample random mini-batch of transitions $(s_t, a_t, r_{t+1}, s_{t+1})$ from D
- Optimize MSE between Q-network and Q-learning targets, e.g.

$$\mathcal{L}(w) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}}\left[\left(r + \gamma \max_{a'} Q(s', a', w) - Q(s, a, w)\right)^2\right]$$

This breaks the similarity of subsequent training samples, which otherwise might drive the network into a local minimum.

Stable Deep RL (1): Experience Replay

Problem: approximation of Q-values using non-linear functions is not stable

Solution: