Deep Learning
Lab17: Deep RL

Pin-Yu Wang & Datalab
Outline

• From RL to Deep RL
• Deep Q-Network
  • Naïve Algorithm(TD)
  • Experience Replay
  • Delayed Target Network
  • Complete Algorithm
  • Implementation
• Assignment
Outline

• From RL to Deep RL
  • Deep Q-Network
    • Naïve Algorithm(TD)
    • Experience Replay
    • Delayed Target Network
    • Complete Algorithm
    • Implementation

• Assignment
From RL to Deep RL

• (Tabular) RL
  • $Q$-learning:
    \[
    Q^*(s, a) \leftarrow Q^*(s, a) + \eta [(R(s, a, s') + \gamma \max_{a'} Q^*(s', a')) - Q^*(s, a)]
    \]
  • It requires a large table to store $Q^*$ values in realistic environments with large state/action space.
    • Flappy bird: $O(10^5)$, Tetris: $O(10^{60})$, Automatic car: ???
  • Hard to visit all $(s, a)'s$ in limited training time.
  • Agents must derive efficient representations of the environment from high-dimensional inputs and use these to generalize past experience to new situations.
Outline

• From RL to Deep RL

• Deep Q-Network
  • Naïve Algorithm(TD)
  • Experience Replay
  • Delayed Target Network
  • Complete Algorithm
  • Implementation

• Assignment
Deep Q-Network

• To learn a function $f_{Q^*}(s, a; \theta)$ that approximates $Q^*(s, a)$
  • Trained by a small number of samples.
  • Generalize to unseen states/actions.
  • Smaller $\theta$ to store.

![Diagram of Deep Q-Network]

$s \rightarrow f \rightarrow Q(s, a^{(1)}) \rightarrow Q(s, a^{(2)}) \rightarrow Q(s, a^{(3)})$
Deep Q-Network

- Naïve Algorithm(TD)
  1. Take action $a$ from $s$ using some exploration policy $\pi'$ derived from $f_{Q^*}$ (e.g. $\varepsilon$-greedy).
  2. Observe $s'$ and reward $R(s, a, s')$, and update $\theta$ using SGD:

$$\theta \leftarrow \theta - \eta \nabla_{\theta} C,$$

where

$$C(\theta) = [(R(s, a, s') + \gamma \max_{a'} f_{Q^*}(s', a'; \theta)) - f_{Q^*}(s, a; \theta)]^2$$

- Recall the $Q$-learning update formula:

$$Q^*(s, a) \leftarrow Q^*(s, a) + \eta [(R(s, a, s') + \gamma \max_{a'} Q^*(s', a')) - Q^*(s, a)]$$
Deep Q-Network

• However, the naïve TD algorithm diverges due to:
  1. Samples are correlated (violates i.i.d. assumption of training examples).
  2. Non-stationary target ($f_{Q^*}(s', a'; \theta)$ changes as $\theta$ is updated for current $a$).

• As a result, the Deep Q-Network applies two stabilization techniques to solve each problem respectively:
  1. Experience Replay
  2. Delayed Target Network
Outline

• From RL to Deep RL

• Deep Q-Network
  • Naïve Algorithm(TD)
  • Experience Replay
  • Delayed Target Network
  • Complete Algorithm
  • Implementation

• Assignment
Deep Q-Network

• Experience Replay
  • To remove the correlations in the observation sequence.
  • Use a replay memory $\mathbb{D}$ to store recently seen transitions $(s, a, r, s')$'s.
  • Sample a mini-batch from $\mathbb{D}$ and update $\theta$.
    • The sample from the mini-batch is not a sequence now.
Outline

• From RL to Deep RL

• Deep Q-Network
  • Naïve Algorithm(TD)
  • Experience Replay
  • Delayed Target Network
  • Complete Algorithm
  • Implementation

• Assignment
Deep Q-Network

• Delayed Target Network
  • To avoid chasing a moving target.
  • Set the target value to the output of the network parameterized by old $\theta^-$.
  • Update $\theta^- \leftarrow \theta$ every $K$ iterations.

❖ Update formula of naïve TD algorithm:

$$C(\theta) = [(R(s, a, s') + \gamma \max_{a'} Q^*(s', a'; \theta)) - Q^*(s, a; \theta)]^2$$

❖ Update formula after applying Delayed Target Network:

$$C(\theta) = [(R(s, a, s') + \gamma \max_{a'} Q^*(s', a'; \theta^-)) - Q^*(s, a; \theta)]^2$$
Outline

• From RL to Deep RL

• Deep Q-Network
  • Naïve Algorithm (TD)
  • Experience Replay
  • Delayed Target Network
  • Complete Algorithm
  • Implementation

• Assignment
Deep Q-Network

• Complete Algorithm
  • Naïve algorithm (TD) + Experience Replay + Delayed Target Network
  • Initialize $\theta$ arbitrarily and set $\theta^- = \theta$. Iterate until converge:
    1. Take action $a$ from $s$ using some exploration policy $\pi'$ derived from $f_Q^*$ (e.g. $\varepsilon$-greedy).
    2. Observe $s'$ and reward $R(s, a, s')$, add $(s, a, R, s')$ to $\mathbb{D}$.
    3. Sample a mini-batch of $(s, a, R, s')$'s from $\mathbb{D}$, do:
       $$\theta \leftarrow \theta - \eta \nabla_\theta C,$$
       where
       $$C(\theta) = [(R(s, a, s') + \gamma \max_{a'} f_Q^*(s', a'; \theta^-)) - f_Q^*(s, a; \theta)]^2$$
    4. Update $\theta^- \leftarrow \theta$ every $K$ iterations.
Outline

• From RL to Deep RL

• Deep Q-Network
  • Naïve Algorithm(TD)
  • Experience Replay
  • Delayed Target Network
  • Complete Algorithm
  • Implementation

• Assignment
Outline

• From RL to Deep RL
• Deep Q-Network
  • Naïve Algorithm(TD)
  • Experience Replay
  • Delayed Target Network
  • Complete Algorithm
  • Implementation

• Assignment
Assignment – state-based DQN

• What you should do:
  • Change the input from stack of frames to game state (as Lab 16).
  • Change the network structure from CNNs to Dense layers.
  • Train the state-based DQN agent to play Flappy Bird.

• Evaluation metrics:
  • Code (Whether the implementation is correct) (50%).
  • The bird is able to fly through at least 1 pipes (50%).

• Requirements:
  • Upload the notebook and html file to google drive, and submit the link to iLMS.
    • Lab17_{student_id}.ipynb
    • Lab17_{student_id}.html
  • Deadline: 2020-12-31(Thur) 23:59.