Session Week 4: DQN + Variants

CS 234

Reinforcement Learning

Feb. 5 - 8, 2019

Overview

- DQN Issues + solutions
- Double DQN
- Prioritized Replay
- Dueling DQN
- Exercises

Review of Deep Q-Learning

Q-Learning with VFA can diverge, two issues:

- correlations between samples
- non-stationary targets

Review of Deep Q-Learning

Q-Learning with VFA can diverge, two issues:

- Correlations between samples
- Non-stationary targets

Solutions:

- Experience Replay (break correlations by sampling from a replay buffer)
- Fixed Q-target (fix the target weights used in the target calculation for multiple updates)

Review of Deep Q-Learning

DQN variants:

- Double DQN (remove the maximization bias)
- Prioritized Replay (sample tuples for update using priority function, which is proportional to DQN error)
- Dueling DQN (separate advantage from value)

Maximization Bias and Double DQN

- (Thrun & Schwartz 1993)
- Consider some set of $Q_{approx}(s, a_1), Q_{approx}(s, a_2), Q_{approx}(s, a_3)$ that differ from the true Q values by some random, but zero-mean noise. Some Q_{approx} are too large, and some are too small.
- Now imagine that the true Q-values for this state are equal for all actions: $Q(s, a_1) = Q(s, a_2) = Q(s, a_3)$
- When we take a max over actions: $\max_a Q_{approx}(s, a)$, we will likely overestimate the true value of state s because the max operator does not preserve the zero mean nature of the errors
- Solution: Double DQN. Choose the "best" action with a Q from one set of samples, and estimate the value of that "best" action with a Q derived from a second set of samples

Comparing Fixed Target + Double DQN

Mnih et al 2015 fixed target TD error:

$$(r + \gamma \max_{a'} \hat{Q}(s', a'; w^-) - \hat{Q}(s, a; w))$$

Double DQN fixed target TD error:

$$(r + \gamma \hat{Q}(s', \arg\max_{a'} \hat{Q}(s', a'; w); w^-) - \hat{Q}(s, a; w))$$

where w^- are old weights

Exercise 1

Why not $(r + \gamma \hat{Q}(s', \arg\max_{a'} \hat{Q}(s', a'; w^-); w) - \hat{Q}(s, a; w))$?

Solution to Exercise 1

One of our goals is to keep our TD target stable for a few steps to minimize “chasing a moving target”. The following expression achieves this:

$$(r + \gamma \hat{Q}(s', \arg\max_{a'} \hat{Q}(s', a'; w)\underline{; w^-)} - \hat{Q}(s, a; \underline{w}))$$

While this next expression does not:

$$(r + \gamma \hat{Q}(s', \arg\max_{a'} \hat{Q}(s', a'; w^-)\underline{; w)} - \hat{Q}(s, a; \underline{w}))$$

In the second equation, we are evaluating the state-action value with the new weights both times.

Prioritized Replay

Consider the following example (from Schaul et al, 2015).

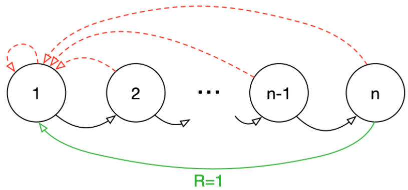


Figure 1: the 'Blind Cliffwalk' example domain: there are two actions, a 'right' and a 'wrong' one, and the episode is terminated whenever the agent takes the 'wrong' action (dashed red arrows). Taking the 'right' action progresses through a sequence of n states (black arrows), at the end of which lies a final reward of 1 (green arrow); reward is 0 elsewhere.

Prioritized Replay

Consider two agents, both performing Q-learning with the same replay buffer. The first agent samples transitions uniformly at random while the second invokes an oracle to prioritize transitions. The results look like:

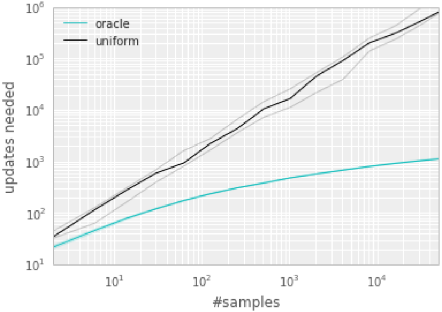


Figure 2: the x-axis is the number of samples in replay buffer; y-axis is the number of updates needed. We see an **exponential** speedup from replaying from an oracle.

Prioritized Replay

- Priority of a tuple i is proportional to DQN error

$$p_i = \left| r + \gamma \max_{a'} Q'(s_{i+1}, a'; w^-) - Q(s_i, a_i; w) \right|$$

- Update p_i every update
- $p_i = 0$ for new tuples
- Stochastic prioritization (adjusting α):

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

Exercise 2

What is one limitation of using the original experience replay buffer formulation in DQN?

Solution to Exercise 2

What is one limitation of using the original experience replay buffer formulation in DQN?

Solution: the replay buffer does not differentiate informative transitions and it always overwrites the buffer with the recent transitions (that's why prioritized experience replay is better).

Dueling DQN

Define an advantage function: $A(s, a) = Q(s, a) - V(s)$
Intuition: features useful for making decisions might be different from those for evaluating the “value” of a state. Therefore, separating the advantage from the value V can be helpful and make the training easier.

Dueling DQN

Unidentifiablility of $Q(s,a) = V(s) + A(s,a)$: given Q, we cannot recover V and A uniquely (why?)

Two options:

- Force $A(s,a) = 0$ for the current best action

$$\hat{Q}(s,a;w) = \hat{V}(s;w) + \left(\hat{A}(s,a;w) - \max_{a' \in \mathcal{A}} \hat{A}(s,a';w)\right)$$

- Alternatively, use mean as the baseline

$$\hat{Q}(s,a;w) = \hat{V}(s;w) + \left(\hat{A}(s,a;w) - \frac{1}{|\mathcal{A}|} \sum_{a'} \hat{A}(s,a';w)\right)$$

Comparing Types of Q Learning

- Tabular Q learning
  $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left((r_t + \gamma \max_{a'} Q(s_{t+1}, a')) - Q(s_t, a_t)\right)$
- Q learning with function approximation
  Minimize a loss, e.g. the squared error, between one approximate Q-function and a better approximate Q-function by doing gradient descent

$$L = (r + \gamma \max_a \hat{Q}(s_{t+1}, a; w) - \hat{Q}(s_t, a_t; w))^2$$

$$w \leftarrow w + \alpha \nabla_w (L)$$

  - Linear Features
    The gradient is simple function of the feature function, x

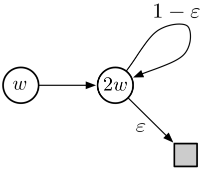
$$w \leftarrow w + \alpha(r + \gamma \max_a \hat{Q}(s_{t+1}, a; w) - \hat{Q}(s_t, a_t; w))x(s, a)$$

$$= w + (\text{learning rate})(\text{TD error})(\text{feature function})$$

  - Neural Networks (DQN) The gradient expression depends on the network structure

$$w \leftarrow w + \alpha(r + \gamma \max_a \hat{Q}(s_{t+1}, a; w) - \hat{Q}(s_t, a_t; w))\nabla_w NN(s, a)$$

Exercise 3: Linear Value Function Approximation, Example 11.1 (S&B)



There is one action in each of the two states, and the gray box is terminal. The reward is zero on all transitions. We use a linear value function approximator where there is one weight, w, and the feature function f is defined as such: $f(s_1) = 1, f(s_2) = 2$, making the estimated value of the first state w and the estimate value of the second state $2w$. Do a weight update using the squared error between the estimated value and the expected one-step return, finding the least-squares value for the new weight w_{k+1}. Will the value function approximation converge to the true value function, $V(s_1) = 0, V(s_2) = 0$?

Solution to Exercise 3

$$w_{k+1} = \arg\min_{w \in \mathbb{R}} \sum_{s \in \mathcal{S}} \left(\hat{V}(s, w) - \mathbb{E}_\pi \left[R_{t+1} + \gamma \hat{V}(s_{t+1}, w_k) \right] \right)^2$$

$$= \arg\min_{w \in \mathbb{R}} \ (w - \gamma 2 w_k)^2 + (2w - (1 - \epsilon)\gamma 2 w_k)^2$$

$$= \frac{6 - 4\epsilon}{5} \gamma w_k$$

The sequence w_k diverges when $\gamma > \frac{5}{6-4\epsilon}$ and $w_0 \neq 0$

Key Takeaway: VFA may not converge.
Note: This is not Q-learning because we are estimating V and not Q.