Asynchronous Methods for Deep Reinforcement Learning

Reminders

- Policy is updated in the direction of the gradient (i.e. gradient ascent)

Overview

- Previously thought that combination of RL and DL (=DRL) is inherently unstable
  - Sequence of observations is non-stationary and online updates are strongly correlated
- Multiple methods try to stabilize the learning process
- One of them is experience replay which has drawbacks:
  - On every interaction we learn from a batch of previous interactions $\rightarrow$ memory and computation requirements
  - Requires off-policy learning - updated from data generated by an older policy
- V^π used as baseline to decrease the variance $\rightarrow R_t - b_t \approx A = Q - V$ (= *advantage*)
- No replay memory - learning stabilized by diversity from different actors
  - Allows for on-policy algorithms
  - Diversity comes not only from randomized environment, but also from different initializations and parameters
- Actually presented 4 algorithms:
  - one-step Q-learning (off-policy), one-step Sarsa (on-policy), n-step Q-learning and advantage actor-critic

Key ingredients

- Asynchronously running multiple agents to interact with the environment and update the same targets
- Replaced a "conventional" one-step Q-learning update

  $$L(\theta_i) = \mathbb{E}[Q(s,a;\theta_i) - r - \gamma \max_{a'} Q(s',a';\theta_{i-1})]^2$$

  with an n-step update

  $$L = \mathbb{E}[Q(s,a) - \sum_{k=0}^{n-1} \gamma^k r_{t+k} - \gamma^n \max_{a'} Q(s_{t+n},a')]^2$$

  - This helps propagating the rewards faster
- Subtracting a learned function of the state $b(s)$ (called *baseline*) from the return reduces variance, while staying **unbiased**
- RMSprop used for optimization - parameters shared across threads

Comments

- Grounds and Kudenko proposed a similar approach in 2008
  - They ran multiple agents and broadcasted updates to significant weight changes from each agent
- In the continous case, since the step is performed after an episode is finished, just running it sequentially shoud work?
- Having a hard time making it work with Keras
- From the notation, not quite clear what is affected by the derivative ∇_θ