Notes: Asynchronous Methods for Deep Reinforcement Learning[1]

esgl Hu

December 14, 2017

1 Review

The simple online RL algorithms with deep neural networks was fundamentally unstable.

The sequence of observed data encountered by an online RL agent is non-stationary, and online RL updates are strongly correlated.

The experience replay has several drawbacks: it uses more memory and computation per real interaction; and it requires off-policy learning algorithms that can update from data generated by an older policy.

The loss function of one-step Q-learning is

$$L_i(\theta_i) = \mathbb{E}(r + \gamma \cdot max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i))^2 \tag{1}$$

where s' is the state encounted after state s. One drawback of using one-step method is that obtaining a reward r only directly affects the value of the state action pair s, a that led to the reward. The values of other state action pairs are affected only indirectly througt the updated value $Q(s, a)$. This can make the learning process slow since many updates are required the propagate a reward to the relevant preceding states and actions. One way of propagating rewards faster is by using n-step returns. In n-step Q-learning, $Q(s, a)$ is updated toward the n-step return defined as $r_t + \gamma r_{t+1} + \cdots + \gamma^{n-1} r_{t+n-1} + \max_a \gamma^n Q(s_{t+n}, a)$.

Policy-based model-free methods directly parameterize the policy $\pi(a|s; \theta)$ and update the parameters θ by performing, typically approximate, gradient ascent on $\mathbb{E}[R_t]$. For example, standard **REINFORCE** updates the policy parameters θ in the direction $\nabla_\theta \log \pi(a_t|s_t; \theta) R_t$, which is an unbiased estimate of $\nabla_\theta \mathbb{E}[R_t]$. It is possible to reduce the variance of this estimate while keeping it unbiased by substracting a learned function of the state $b_t(s_t)$, known as a baseline from the return. The resulting gradient is $\nabla_\theta long \pi(a_t|s_t; \theta)(R_t - b_t(s_t))$.

Algorithm 1 Asychronous one-step Q-learning - pseudocode for each actor-learner thread.

// Assume global shared θ, θ^{-}, and counter $T = 0$.
Initialize thread step counter $t \leftarrow 0$
Initialize target network weights $\theta^{-} \leftarrow \theta$
Initialize network gradients $d\theta \leftarrow 0$
Get initial state s
repeat
  Take action a with ϵ-greedy policy based on $Q(s, a; \theta)$
  Recieve new state $s^{'}$ and reward r
  $y = \begin{cases} r & \text{for terminal } s^{'} \\ r + \gamma max_{a^{'}} Q(s^{'}, a^{'}; \theta^{-}) & \text{for non-terminal } s^{'} \end{cases}$
  Accumulate gradients wrt : $d\theta \leftarrow d\theta + \frac{\partial(y - Q(s,a;\theta))^2}{\partial\theta}$
  $s = s^{'}$
  $T \leftarrow T + 1$ and $t \leftarrow t + 1$
  if T mod $I_{target} == 0$ **then**
    Update the target network $\theta^{-} \leftarrow \theta$
  end if
  if t mod $I_{asyncUpdate} == 0$ or s is terminal **then**
    Perform asynchronous update of θ using $d\theta$.
    Clear gradients $d\theta \leftarrow 0$.
  end if
until $T > T_{max}$

2 Algorithm

References

[1] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. CoRR, abs/1602.01783, 2016.