Adaptive Learning Rates for Multi-Agent Reinforcement Learning

Colin Acker

October 2023

1 Notes

For MADDPG, Deep Deterministic Policy Gradient extended to Multi-Agent systems, the agents share a single reward. Each agent learns an actor π_i and a single shared critic. The critic takes $\vec{a} = (a_1, a_2 ... a_n)$ and $\vec{o}$, the observation vector, as input. The output of the critic is the Q-value. The critic, parameterized by ϕ, is trained by minimizing the TD-error δ

$$\mathbb{E}_(\vec{o}, \vec{a}, r, \vec{o}')[(Q(\vec{o}, \vec{a}) - y)^2]$$

$$\text{Where } y = r + \gamma Q^-(\vec{o'}, \pi_i^-(o_i'))$$

Q^- is the target critic, π_i^- is the target actor, and each $(\vec{o}, \vec{a}, r, \vec{o}')$ is from the replay buffer. The gradient of Q with respect to θ_i is

$$\frac{\partial Q(\vec{o}, \vec{a})}{\partial a_i} \frac{\partial a_i}{\partial \theta_i}$$

The learning rates of each actor critic and the critic will be l_{a_i} and l_c. If we leave the critic alone, we can approximate the gain in the Q-value contributed solely by the actors, using Taylor approximation.

$$\Delta Q = Q(\vec{o}, \vec{a} + \Delta \vec{a}) - Q(\vec{o}, \vec{a})$$

After some math, we get,

$$\Delta Q \approx \vec{l}_a \cdot \frac{\partial \vec{Q}}{\partial \theta} \frac{\partial Q^T}{\partial \theta}$$

So, the largest ΔQ is obtained when the direction of $\vec{l_a}$ is the same direction as $\frac{\partial \vec{Q}}{\partial \theta} \frac{\partial Q^T}{\partial \theta}$. This is basically gradient ascent on $\vec{l_a}$ to increase the Q-value. This only works when the critic is static, but in MADDPG, the critic and actor are trained simultaneously. So now,

$$\Delta Q = Q(\phi + \Delta \phi, \vec{o}, \vec{a} + \Delta \vec{a}) - Q(\phi, \vec{o}, \vec{a})$$

After some math, we get,

$$\Delta Q \approx \vec{l}_a \cdot \frac{\partial \vec{Q}}{\partial \theta} \frac{\partial Q^T}{\partial \theta} - l_c \frac{\partial \delta}{\partial \phi} \frac{\partial Q^T}{\partial \phi}$$

When the critic's update causes a large change in the Q-value, the critic is leading the learning. This is because the actor's learning is determined by the critic. When the critic is leading the learning, we should assign it a high learning rate. The actors struggle with a quickly changing critic, so we should assign the actors a low learning rate. On the other hand, when the critic has hit a plateau, we should increase the actor's learning rate. The learning rate for the critic, l_c is found by updating l_c using a clip function and many different hyperparameters.

$$l_c = \alpha l_c + (1 - \alpha) l \cdot \text{clip}(|\frac{\partial \delta}{\partial \phi} \frac{\partial Q^T}{\partial \phi}|/m, \epsilon, 1 - \epsilon)$$

$$\mu = l - l_c$$

I believe this is the most important result of the paper. The paper goes into details on how to tune these hyperparameters. In the paper, they called the algorithm AdaMa, which updates l_c and l_{a_i} using the equations above. They tested it on 4 different scenarios, all which I think would be good to test on. The key find during testing was adaptively updating l_c and l_{a_i}, works faster than a fixed LR.

1.1 Algorithms

Algorithm 1 AdaMa on MADDPG

Initialize critic network ϕ, actor networks θ_i, target networks, and the replay buffer $\mathcal{D}$.
Initialize the learning rates l_c and $\vec{l_a}$.
for episode = $1, \ldots, \mathcal{M}$ **do**
  for $t = 1, \ldots, \mathcal{T}$ **do**
    Select action $a_t^i = \pi_i(o_t^i) + \mathcal{N}_t^i$ for each agent i
    Execute action a_t^i, obtain reward r_t, and get new observation o_{t+1}^i for each agent i
    Store transition $(\vec{o}_t, \vec{a}_t, r_t, \vec{o}_{t+1})$ in $\mathcal{D}$
  end for
  Sample a random minibatch of transitions from $\mathcal{D}$
  Adjust l_c and $\|\vec{l_a}\|$ by (2).
  Adjust $\vec{l_a}$ by (1) (first order) or (3) (second order).
  Update the critic ϕ by $\phi = \phi - l_c \frac{\partial \delta}{\partial \phi}$.
  Update the actor θ_i by $\theta_i = \theta_i + l_{a_i} \frac{\partial Q(\vec{o}, \vec{a})}{\partial a_i} \frac{\partial a_i}{\partial \theta_i}$ for each agent.
  Update the target networks.
end for