Multi Agent Reinforced Learning

BHAGYA VINOD RANA

① Each Agent has observation space and continous Action space.

Each Agent has 3 components ① Actor-network - Uses local observations for deterministic actions
② target actor-network - identical functionality for training stability
③ critic network - uses joint state action pair for estimate Q pairs

As critic learns the joint Q-value function over time,
it sends appropriate Q-value approximations to the actor to help training.

Ⓐ At each timestampt, agent stores following transition

$$(x, x', a_1, a_2, a_3, \ldots a_N, r_1, r_2, r_3, \ldots r_N)$$

we store join state, next join state and each of agent's received rewards.

Ⓑ Critic Updates

To update an agent's centralized critic, we use one-step lookahead TD error.

$$\mathcal{L}(\theta_i) = \mathbb{E}_{x,a,r,x'}\left[\left(Q_i^{\mu}(x, a_1, \ldots, a_N) - y\right)^2\right]$$

$$y = r_i + \gamma\, Q_i^{\mu'}(x', a_1', \ldots a_N')\Big|_{a_j' = \mu_j'(o_j)}$$

μ = actor

Ⓒ Actor Updates

$$\nabla_{\theta_i} J(\mu_i) = \mathbb{E}_{x,a\sim D}\left[\nabla_{\theta_i \mu_i}(a_i|o_i)\, \nabla_{a_i} Q_i^{\mu}(x, a_1, \ldots a_N)\Big|_{a_i = \mu_i(o_i)}\right]$$

We take gradient with respect to actor's parameters using central critic to guide us.

Ⓓ Policy Inference and Policy Ensembles

We can use probabilistic network & maximize the log probability of outputting another agent's observed action

$$\mathcal{L}(\theta_i^j) = -\mathbb{E}_{o_j, a_j}\left[\log \hat{\mu}_i^j(a_j|o_j) + \lambda H(\hat{\mu}_i^j)\right]$$

Loss function for i^{th} agent estimating j^{th} agent's policy with an entropy regularizer

Q value target (where $\hat{y}$ is)

$$\hat{y} = r_i + \gamma\, Q_i^{\mu'}(x', \hat{\mu}_i'^1(o_1), \ldots, \hat{\mu}_i'(o_i), \ldots \hat{\mu}_i'^N(o_N))$$

[We have removed the assumption that agent's knows each other policies]