Policy-Based Reward Estimation for Expert-Driven Social Learning in Multi-Agent Systems

Nikoo Paknia, Soleiman B.Ganji

January 2025

1 Introduction

In multi-agent reinforcement learning, leveraging the experiences of other agents can significantly enhance an agent's learning efficiency and performance. This project focuses on incorporating social learning by identifying and learning from the most effective agent (expert) in the environment. To achieve this, we estimate the policies of all agents and normalize them against our agent's policy to calculate a factor normalizing factor. By multiplying this factor by our agent's average reward for each a and s, we estimate the reward our agent could achieve by adopting the reference agent's policy. The agent with the maximum normalized reward is then selected as the expert, whose experience is utilized to boost our agent's learning, leading to improved decision-making and accelerated convergence.

2 Problem Setting

2.1 Expert Agent

The expert agent will be a pre-trained Deep-SARSA agent which follows a close-to-optimal policy.

2.2 Policy Estimation

Here, we aim to learn the policy of the other agents $\pi(a|s)$, which represent the probability distribution over actions a given a state s. The data is obtained from a replay buffer containing tuples (s, a) of states and actions. We will use the estimated policy of other agents to estimate an average reward R_{hat} for the $agent_i$ based on our policy and experiences.

Neural Network Architecture

We use a neural network parameterized by θ, denoted as $\pi_\theta(a|s)$. The network:

- Takes the state s as input.
- Outputs the parameters of the probability distribution over actions a. For instance, in a Gaussian policy, the output could be the mean $\mu_\theta(s)$ and standard deviation $\sigma_\theta(s)$.

Objective Function

The objective is to maximize the likelihood of actions taken by the agent in the states observed in the replay buffer. This can be formulated as minimizing the negative log-likelihood:

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\log \pi_\theta(a_i|s_i)$$

where:

- N is the number of samples in the replay buffer.
- (s_i, a_i) are the state-action pairs from the replay buffer.
- $\pi_\theta(a_i|s_i)$ is the probability of action a_i under the policy π_θ given the state s_i.

Gaussian Policy Example

If we assume a Gaussian policy, the probability distribution over actions is:

$$\pi_\theta(a|s) = \frac{1}{\sqrt{2\pi\sigma_\theta(s)^2}}\exp\left(-\frac{(a-\mu_\theta(s))^2}{2\sigma_\theta(s)^2}\right)$$

The loss function becomes:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{(a_i-\mu_\theta(s_i))^2}{2\sigma_\theta(s_i)^2} + \log\sigma_\theta(s_i) + \frac{1}{2}\log(2\pi)\right)$$

Training Process

The training process involves:

1. Sampling mini-batches of state-action pairs (s, a) from the replay buffer.
2. Computing the loss $\mathcal{L}(\theta)$ using the chosen policy distribution.
3. Backpropagating the gradients of the loss to update the neural network parameters θ.

Output of the Network

The neural network outputs:

- For Gaussian policy: $\mu_\theta(s)$ and $\sigma_\theta(s)$.
- For other distributions (e.g., categorical): the parameters of the respective probability distribution (e.g., logits for a softmax output).

2.3 Selecting the Reference Agent

The reference agent is the agent with maximum estimated reward R_{hat}, where:

$$R_{hat} = \frac{1}{T} \sum_{t=1}^{T} \frac{\pi_i(a_t|s_t)}{\pi_b(a_t|s_t)} \cdot R_{s,a}$$

2.4 Social Learning

After selecting the reference agent, we will use the following loss function to guide the training of our network:

$$J(\theta) = \mathbb{E}[r \mid \pi] + cF(\pi, \pi_{\text{ref}})$$

where:

$$F(\pi, \pi_{\text{ref}}) = D_{\text{KL}}(\pi \parallel \pi_{\text{ref}}) = \sum_a \pi(a) \log \frac{\pi(a)}{\pi_{\text{ref}}(a)}$$

3 Going Further

After examining the results of the previous section, it might be interesting to also identify a negative reference agent, that is, an agent that is doing very poorly based on our policy. After finding the negative reference agent, we can aggregate it into our loss function like this:

$$J(\theta) = \mathbb{E}[r \mid \pi] + cF(\pi, \pi_{\text{ref+}}) - cF(\pi, \pi_{\text{ref-}})$$

4 Pseudo-Code

Algorithm 1 The Social Learning Algorithm

Initialize c, π_b and $\pi_i \forall i \in Agents$
for each trial t **do**
    if t mod $k = 0$, sample mini-batches of state-actions and train the policy estimators for all agents
    Sample $A_t \sim \pi_b$ and observe reward R_t, and other agents' actions
    Calculate average reward $\mathbb{E}_{\pi_b}[r_t]$
    Estimate $\pi_i \forall i \in Agents$
    Select the agent which maximizes $\rho\bar{R}$ as the reference agent
    Set the loss function to $J(\theta) = \mathbb{E}[r \mid \pi] + cF(\pi_b, \pi_{\text{ref}})$ and backpropagate
end for