Soft n Actor-Critic Reinforcement Learning Algorithm

James Thompson

April 23, 2025

1 Introduction

The idea of the Soft n Actor-Critic (SnAC) is to use multiple actors which are guided by a single critic. Itr is based off SAC. This single critic is used to help evaluate the potential actions that the ensemble of actors offer. The use of the ensemble of actors allows for each actors to explore and specilize in different areas of the state space. This idea takes rough inspiration from the idea of Mixute of expert models in supervised learning.

2 The SnAC Algorithm

SnAC is very similiar to the Soft Actor-Critic algorithm. I uses a simliar structure as well as similar objective functions. The main difference is that SnAC uses multiple actors that have their polices aggregated to form a single action selection.

As it is a variant of SAC it uses the same objective functions for both critic training (Equation 1) and the entropy coefficient (Equation 2). There is a slight difference however as the objective functions are using the previously used actor.

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(a_t, s_t)\left(Q_\theta(s_t, a_t) - \left(r_t + \gamma\left(Q_{\bar{\theta}}(s_{t+1}, a_{t+1}) - \alpha \log\left(\pi_{\phi_{i_\pi}}(a_{t+1}|s_{t+1})\right)\right)\right)\right) \tag{1}$$

$$\nabla_\alpha J(\alpha) = -\log \pi_t(a_t \mid s_t) - \mathcal{H} \tag{2}$$

What is different is the way that the actors are trained and the objective function. The actors are trained using an objective function that

To ensure that the policies do not converge too closely to each other, we introduce a diversity-promoting term in the policy objective function. The objective function for each policy π_{ϕ_i} is defined as:

$$J_\pi(\phi_i) = \mathbb{E}_{s_t \sim \mathcal{D}}\left[\min_{j \in \{1,2\}} Q_{\theta_j}(s_t,) - \alpha_{\text{div}} \sum_{k \neq i} \text{KL}(\pi_{\phi_i} \| \pi_{\phi_k})\right] \tag{3}$$

Here:

- $\min_{j \in \{1,2\}} Q_{\theta_j}(s_t, a_t)$ ensures the policy maximizes the minimum critic estimate which is used in the SAC algorithm.
- α_{div} is a hyperparameter controlling the strength of the diversity term. *Maybe this could be learned in a similar way to the entropy coefficient.*

- $\mathrm{KL}(\pi_{\phi_i} \| \pi_{\phi_k})$ is the Kullback-Leibler divergence between the current policy π_{ϕ_i} and another policy π_{ϕ_k}, encouraging policies to remain distinct.

This objective ensures that each policy maximizes its expected return while maintaining diversity among the ensemble of policies.

Algorithm 1 Soft n Actor-Critic Algorithm

```
Require: Replay buffer size $N$, Minibatch size $K$, discount factor $\gamma$, learning rates $\lambda$, $\lambda_\pi$
    and $\lambda_Q$, update frequency $C$, target smoothing coefficient $\tau$, Updates to Data ratio
    UTD, environemt steps $S$, number of actors $n$, ensemble aggregation function agg,
    policy divergence coefficient $\alpha_{\text{div}}$, neural network function approximator $Q$, neural
    network function approximator $\pi$ and initial entropy coefficient $\alpha$.
Initialize replay memory $\mathcal{D} \leftarrow \emptyset$ with capacity $N$
Initialize policies $\pi_{\phi_i}$ with random weights $\phi_i$ for $i \in \{1, ..., n\}$
Initialize both action-value function $Q_{\theta_i}$ with random weights $\theta_i$
Initialize $\bar{\theta}_i \leftarrow \theta_i$ for $i \in \{1, 2\}$
repeat
    for  $S$ steps do
        Enumerate action options: $a_{t_i} \sim \pi_{\phi_i}(a_t | s_t)$ for $i \in \{1, ..., n\}$
        Choose action $a_t \leftarrow \mathrm{agg}(\{a_{t_1}, ..., a_{t_n}\}, \theta_1, \theta_2)$ and note the used policy index $i_\pi$
        Observe reward $r_t$ and next state $s_{t+1}$
        Store transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$
    end for
    for UTD times do
        Update Q-functions: $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
        Update policy: $\phi_i \leftarrow \phi_i - \lambda_\pi \hat{\nabla}_{\phi_i} J_\pi(\phi_i)$ for $i \in \{1, ..., n\}$
        Update entropy coefficient: $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$
        Update target Q-networks: $\bar{\theta}_i \leftarrow \tau\theta_i + (1 - \tau)\bar{\theta}_i$ for $i \in \{1, 2\}$
    end for
until stopping criteria is met
```

3 Potential Aggregation functions

As all of the ensemble of actors are trained and kept seperate from each other at any state there are n number of potential actions which are provided by the actors.

3.1 Single selection

Using the highestest experected reward then we are just doing a modified version of q-learning. Where all the polices are doing is providing a subset to look at. Alternatively one could simply randomly select on e the actions, however this would remove the potential benefit of having actors "specilize" in differnet areas of the state space. However an inbetween approach could be to do a weighted random selection based on the expected reward of the action.

3.2 Average of actions

For a continouis contorl task the action is a bunch of forces to be applied to the various joints. Therefore we could take the average of the actions by either element wise (average move for each joint) or by taking the geomtric average of the actions.

3.3 Different ways to compute the average vector

When aggregating actions by averaging, there are multiple ways to compute the average vector:

- **Element-wise average:** Compute the average (arithmatic or median) for each dimension of the action vector independently. This is straightforward and ensures that each component of the action is averaged across all actors.
- **Weighted average:** Assign weights to each actor's action by using critic network and compute a weighted average.

4 Other ideas

4.1 Policy aware critic

Currently the critic is a function that just takes in the state and action. It is plausible for a critic to learn quite a general state action value function. However, giving the critic a simple indicator of which policy we would be using would mean that it could learn more spcific values for each policy. This would mean that it would need more data to learn as only the policy which are used can be used to update the critic.

4.2 Using a multi headed actor instead of multiple actors

The actors are all individual polices with there own network. It could be effective instead to use a single multi headed actor. This will speed up training time and provide some bounds on how distinct the policies can be. However my sense could be that the benfit of having "experts" for differnet situations may be lost by forcing the internal repsentation to be the same.

5 Conclusion

The Soft n Actor-Critic (SnAC) algorithm is a simple change to the Soft Actor-Critic algorithm. By using mulitple actors we can allow for differnet actors to specilize in different areas of the state space and provide a more diverse set of actions. The use of of a single critic allows it to learn a general approximation of state action values. The diverse set of actions can be aggregated into a single action using this general critic.