Decentralized ADQN

Andrea Baisero

October 25, 2024

Notation: To avoid excessive symbol overloading, we employ $\hat{Q}$ to denote stateless value models, e.g., $\hat{Q}(\boldsymbol{h}, \boldsymbol{a})$ and $\hat{Q}_i(h_i, a_i)$, and $\hat{U}$ to denote stateful value models, e.g., $\hat{U}(\boldsymbol{h}, s, \boldsymbol{a})$ and $\hat{U}_i(h_i, s, a_i)$.

1 Background: ADQN

Asymmetric DQN (ADQN) is a single-agent asymmetric deep learning RL method that exploits privileged state information by training an auxiliary history-state value model $\hat{U}(h, s, a)$ in conjunction with the standard value model $\hat{Q}(h, a)$. Naturally, $\hat{Q}$ implicitly defines the agent's behavior, while the stateful counterpart $\hat{U}$ is used exclusively as a training construct in order to get more informative[1] values to aid the training of $\hat{Q}$.

Let $\hat{\pi}$ denote the greedy policy with respect to $\hat{Q}$, i.e., $\hat{\pi}(h) \doteq \operatorname{argmax}_a \hat{Q}(h, a)$. Then, the models are trained according to

$$y = r + \gamma \hat{U}(hao, s', \hat{\pi}(hao)) \,, \tag{1}$$

$$\mathcal{L}_{\hat{U}}(h, s, a, r, s', o) = \frac{1}{2} \left(\text{stop}\left[y\right] - \hat{U}(h, s, a) \right)^2 \,, \tag{2}$$

$$\mathcal{L}_{\hat{Q}}(h, s, a, r, s', o) = \frac{1}{2} \left(\text{stop}\left[y\right] - \hat{Q}(h, a) \right)^2 \,. \tag{3}$$

Notably, both models are trained on the same target y, which represents the stateful evaluation of the actions chosen by the greedy policy $\hat{\pi}$. Though the target is the same, the models should coverge to different values due to their different expressiveness, i.e., the stateful model $\hat{U}$ should be able to match the target's variations based on the state, and should be able to model the policy values $\hat{U}(h, s, a) \rightsquigarrow U^{\hat{\pi}}(h, s, a)$, while the stateless model $\hat{Q}$ is unable to do so (not having access to the state), and should implicitly average the state information out, but should still be able to model the policy values $\hat{Q}(h, a) \rightsquigarrow Q^{\hat{\pi}}(h, a)$, which implicitly performs policy improvement by having Q^{π} model the values of its own greedy policy.

[1] The justification for calling stateful values "more informative" is questionable and still a topic of further research.

2 Decentralized DQN (Dec-DQN)

There is an opportunity to employ the ADQN approach for decentralized multi-agent control problems. For now, we can ignore state information and focus on decentralized training; two stateful variants will follow.

Let $\hat{\boldsymbol{\pi}}$ denote the joint greedy policy with respect to the individual values $\hat{Q}_i$, i.e., $\hat{\boldsymbol{\pi}}(\boldsymbol{h}) \doteq \left(\text{argmax}_a \hat{Q}_i(h_i, a)\right)_i$.

$$y = r + \gamma \hat{Q}(\boldsymbol{hao}, \hat{\boldsymbol{\pi}}(\boldsymbol{hao})), \tag{4}$$

$$\mathcal{L}_{\hat{Q}}(\boldsymbol{h}, \boldsymbol{a}, r, \boldsymbol{o}) = \frac{1}{2}\left(\text{stop}\left[y\right] - \hat{Q}(\boldsymbol{h}, \boldsymbol{a})\right)^2, \tag{5}$$

$$\mathcal{L}_{\hat{Q}_i}(\boldsymbol{h}, \boldsymbol{a}, r, \boldsymbol{o}) = \frac{1}{2}\left(\text{stop}\left[y\right] - \hat{Q}_i(h_i, a_i)\right)^2. \tag{6}$$

Dec-DQN employs the same methodology as ADQN: the centralized values $\hat{Q}(\boldsymbol{h}, \boldsymbol{a})$ are used as a training construct to aid the training of the individual values $\hat{Q}_i$. Both models are trained on the same target y, which represents the centralized evaluation of the actions chosen by the (joint) greedy policy induced by $\hat{Q}_i$.

What does this amount to? A fancy centralized variant of independent Q-learning, really. Each individual model $\hat{Q}_i$ is trained to maximize the centralized value $\hat{Q}(\boldsymbol{h}, \boldsymbol{a})$, broadly speaking (though all the individual updates happen jointly and softly/smoothly, so this is probably more an analogy than a formal analysis of the convergence properties).

Connection to Value Factorization Though Dec-ADQN is clearly inspired by the ADQN approach, there are some connections to value factorization methods. Specifically, we do not enforce a strict relationship between centralized and decentralized models, and we do not require IGM to hold, but we still end up employing the decentralized actions to compute the learning targets. Value factorization methods rely on IGM because they start from the (flawed) premise of a centralized objective that maximizes over joint values. If we drop the centralized objective premise, we can still simply employ the decentralized actions to define/compute the learning targets, which is what happens in ADQN (just in relation to full/partial observability, not in relation to centralized/decentralized control).

3 Decentralized ADQN (Dec-ADQN)

Decentralized DQN is easily exteded to a stateful variant by directly employing a stateful cetralized model $\hat{U}(\boldsymbol{h}, s, \boldsymbol{a})$. Motivation and theoretical properties are similar.

$$y = r + \gamma \hat{U}(\boldsymbol{hao}, s', \hat{\boldsymbol{\pi}}(\boldsymbol{hao})) \tag{7}$$

$$\mathcal{L}_{\hat{U}}(\boldsymbol{h}, s, \boldsymbol{a}, r, s', \boldsymbol{o}) = \frac{1}{2}\left(\text{stop}\left[y\right] - \hat{U}(\boldsymbol{h}, s, \boldsymbol{a})\right)^2, \tag{8}$$

$$\mathcal{L}_{\hat{Q}_i}(\boldsymbol{h}, s, \boldsymbol{a}, r, s', \boldsymbol{o}) = \frac{1}{2}\left(\text{stop}\left[y\right] - \hat{Q}_i(h_i, a_i)\right)^2. \tag{9}$$

4 Cascading Decentralized ADQN (Cascading Dec-ADQN)

Finally, there's perhaps potential for a cascading variant that trains two different types of centralized values; a stateful one $\hat{U}(\boldsymbol{h}, s, \boldsymbol{a})$ and a stateless one $\hat{Q}(\boldsymbol{h}, \boldsymbol{a})$. The motivation for including this intermediate stateless step is that it be the one to deal with integrating state information, so that the individual values can focus on integrating only the joint histories and actions.

$$y_{\hat{U}} = r + \gamma \hat{U}(\boldsymbol{hao}, s', \hat{\boldsymbol{\pi}}(\boldsymbol{hao})), \tag{10}$$

$$\mathcal{L}_{\hat{U}}(\boldsymbol{h}, s, \boldsymbol{a}, r, s', \boldsymbol{o}) = \frac{1}{2}\left(\text{stop}\left[y_{\hat{U}}\right] - \hat{U}(\boldsymbol{h}, s, \boldsymbol{a})\right)^2, \tag{11}$$

$$\mathcal{L}_{\hat{Q}}(\boldsymbol{h}, \boldsymbol{a}, r, \boldsymbol{o}) = \frac{1}{2}\left(\text{stop}\left[y_{\hat{U}}\right] - \hat{Q}(\boldsymbol{h}, \boldsymbol{a})\right)^2, \tag{12}$$

$$y_{\hat{Q}} = r + \gamma \hat{Q}(\boldsymbol{hao}, \hat{\boldsymbol{\pi}}(\boldsymbol{hao})), \tag{13}$$

$$\mathcal{L}_{\hat{Q}_i}(\boldsymbol{h}, \boldsymbol{a}, r, \boldsymbol{o}) = \frac{1}{2}\left(\text{stop}\left[y_{\hat{Q}}\right] - \hat{Q}_i(h_i, a_i)\right)^2. \tag{14}$$

Other Cascading Variants Instead of cascading values in the order of $\hat{U}(\boldsymbol{h}, s, \boldsymbol{a}) \to \hat{Q}(\boldsymbol{h}, \boldsymbol{a}) \to \hat{Q}_i(h_i, a_i)$ there can be other ways to implement value cascading, e.g., $\hat{U}(\boldsymbol{h}, s, \boldsymbol{a}) \to \hat{U}_i(h_i, s, a_i) \to \hat{Q}_i(h_i, a_i)$, or even mixures of the two.