Progress report: Hanabi

Xiuhan Wang, Jing Wei

December 6, 2020

1 Experimental Results

We cloned the github repository https://github.com/facebookresearch/hanabi_SAD of the open-sourced code of Hanabi SAD with other-play (Hu et al., 2020) and tried to run it. Since it is our first time to do a project, we spent a lot of time on dealing with environmental settings. We first tried to run it on our local machines using cpu, but there appeared to be a lot of problems, including pytorch versions and github repository versions (some repositories have been updated, but it seems the updated version can hardly be compiled successfully). Fortunately, though having spent a lot of time, we managed to solve them and made it run successfully on our machine.

After making it run successfully on our local machines, we rented some GPUs to train it. Because of the limitation of our computation resources, the results haven't converged yet and are worse than we expected. Table 1 is the scores for the models in the first 500 epoches:

epoch	mean score	s.e.m.	perfect ratio
100	17.084	0.037	0.000
200	17.116	0.035	0.001
300	17.160	0.035	0.001
400	17.191	0.035	0.002
500	17.245	0.033	0.003

Table 1: Performances after every 100 more epoches is trained.

The trained model is Simplified Action Decoder (SAD) built on Independent Q-Learning (IQL). The number of players is 2.

Wondering why the performance seems worse than that in the original paper, we reread the original paper and code more carefully, we found out that only when trained about 1 million epoches can the model converge! However, with our resources, it takes almost one day to train 500 epoches.

Further more, the model in the paper is SAD based on VDN, which converges faster and better than IQL. All these reasons made our model perform worse. And it seems extremely hard to reproduce the results of the paper.

We also evaluated the models trained by the paper authors(Hu & Foerster, 2020). We focus on 2-player settings. Table 2 compares the performances under different training settings:

method	mean score	s.e.m.	perfect ratio
IQL	23.773	0.017	0.436
VDN	23.897	0.018	0.487
SAD	23.980	0.018	0.531
SAD + Aux	24.015	0.018	0.544

Table 2: Comparing performances under different training settings.

We can observe that the model trained by SAD + Auxiliary tasks is the best. At the beginning epoches SAD may perform worse than the other settings, but when the model nearly converges, SAD is at a great advantage.

2 Theoretical Ideas

Focusing on the problem in the zero-shot coordination setting that agents are unable to coordinate on how to bread symmetries, Hu et al. (2020) proposed "other-play" to make the model invariant to symmetries in the underlying MDP.

The symmetries in the MDP can be described as a permutation group over states $\mathcal{S}$, observations $\mathcal{O}$ and actions $\mathcal{A}$ that keeps the transition function P, the reward R and the observation function O invariant. What OP results is essentially a model that doesn't break this symmetry. We can achieve this goal more straightforwardly by designing the networks to be symmetric according to this permutation group.

For example, in Hanabi, there is a S_5 symmetry on 5 colors of the cards. Then the network should comform to S_5, satisfying:

- For every neuron h in the network, there should also be neurons $\phi(h)$ for all $\phi \in S_5$.
- h and $\phi(h)$ should have the same bias. If h takes its input from a neuron g with weight α, then $\phi(h)$ takes input from $\phi(g)$ with the same weight.

Hence, we see neurons can be seen as devided into equivalence classes $S_5 h = \{\phi(h) \mid \phi \in S_5\}$, where they share a same set of weights and bias among the same class. The sizes of these equivalent classes are at most 120.

We construct a network in which neurons are named by $h_{t,i,\tau}$ where t is the number of the layer, and $\tau \in T_{t,i}$ where $T_{t,i}$ is a set that S_5 can act on. Let $\phi(h_{t,i,\tau}) = h_{t,i,\phi(tau)}$ and this will meet the conditions.

This way of constructing a network apply to various types of neurons including LSTM cells.

3 Further Directions

Next we have to read the codes more carefully to figuer out the structure of the code and where and how exactly do those various arguments make differences. This might be hard since there are no manual and barely any comments in the code, which makes the code hard to understand.

After that we can implement our ideas proposed in the theoretical section above. And equipted with that knowledge, we believe we can train some more models to get a more reasonable result for a baseline. In order to see the differences more clearly, we may slightly extend the number of epoches to train if time allows. And we are going to restrict the problem to some settings we interest most (for example, we shall focus on 2-player settings) and try to figuer out something more specific.

If the experimental tasks are still too hard, we may change our focus on theoretical ideas. We may read more papers about Hanabi, understand their ideas more clearly, and think about why such settings have better performance than others.

References

Hu, H., & Foerster, J. N. (2020). Simplified action decoder for deep multi-agent reinforcement learning. In Iclr 2020 : Eighth international conference on learning representations.

Hu, H., Lerer, A., Peysakhovich, A., & Foerster, J. (2020). “other-play” for zero-shot coordination. arXiv preprint arXiv:2003.02979.