Submission description

1 General notes

Please note that the metric that will be used for evaluation has not been clearly specified. The webpage indicates that mean per episode will be used, whereas the evaluation script calculates the mean per step. For a single episode they are related by the following equality:

$$E[\sum_i r_i] = E[\hat{R}S]$$

where r_i denotes reward in step i, $\hat{R}_i$ denotes mean reward per step and S denotes the number of steps. Since S depends on policy, the optimal behavior for mentioned objectives is different. We train the agent to maximize reward per episode.

2 Description of the used method

For this challenge we used an agent trained with DoubleDQN[1] with dueling architecture[2]. The state is represented as three dimensional board where width and height represent board coordinates and depth represents the type of entity. Every entity is represented by one and all other fields are filled with zeros.

To make the task simpler we utilize the symmetries in the problem by rotating the board in such a way so the pig is always in upper right corner. This is done by simple transformation of coordinates. This also requires adding additional bit of information to the state which indicates the parity of rotation, since the exits are not invariant w.r.t this transformation.

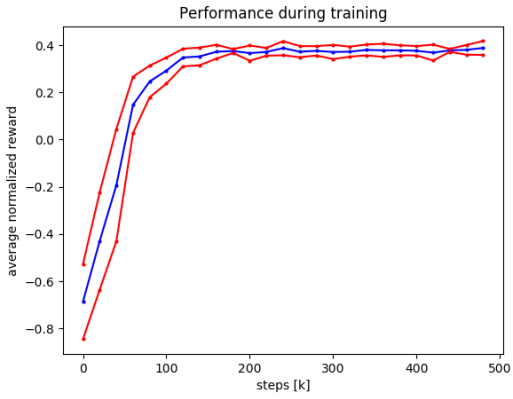
Additionally, we add ten last distances of an opponent to the target and the counter of actions executed to the state to make the task fully observable. Both of these values are normalized to one.

All experiments are carried on a implemented simulator and then evaluated in real environment. To force the agent to learn to catch the target we train it with 0.9 probability of playing with focused agent. It might seem strange that we train and evaluate the agent on different environments but we want the agent to explore the gameplay with cooperating agent much better and we want the updates of parameters for this type of gameplay to dominate. It could be also the case that when the proportion of gameplay with random agent is too large then the optimization reaches local optimum and cannot escape it. Also, learning to escape from board when the second agent is not getting closer to the target is relatively simple and does not require many examples. Test results obtained on simulator when the agent is trained in this way are better.

Lastly, the optimal policy should recognize the type of an agent and carry out an appropriate sequence of actions for both agents, thus any proportion should lead to the same optimal policy. However, for majority of them the learning process might be more difficult.

3 Results

The learning is done with the use of chainerrl library. Every 20000 steps the exploration is stopped and the agent is evaluated for 1000 episodes. Results averaged over 4 runs are depicted below.



[h!]

Figure 1: The normalized mean reward per episode during training.

The results obtained during evaluation on simulator (with probability of focused agent set to 0.75) where equal to 0.3 normalized reward per episode and 1.3 reward per step. This was calculated for 10000 test episodes. Evaluation on the real environment seems

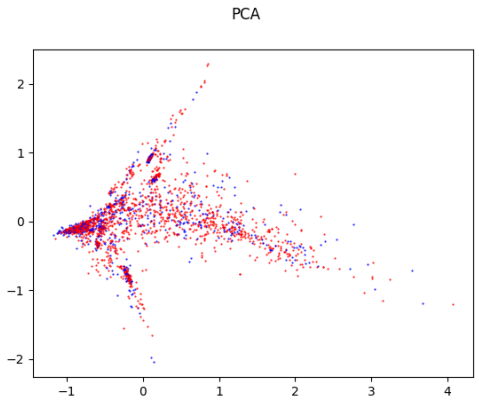


Figure 2: PCA reduction applied to learned representation of state.

to provide the results which are very strongly correlated.

3.1 Learned representation

It is natural to ask whether an value based algorithm can learn an appropriate value function that differentiates both opponents. In order to investigate that we run the trained network on the game for 1000 steps and store the representation of state obtained at the last hidden layer. Linear combinations of these values are learned $Q(s, a)$ values. We performed dimensionality reduction on this data. The results are on figure 2.

It is visible that PCA indicates that there are separete states of the game; we think that moving to the left of the picture corresponds to escaping the map and staying at right corresponds to catching the target.

References

[1] Hado van Hasselt, Arthur Guez, David Silver : Deep Reinforcement Learning with Double Q-learning

[2] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, Nando de Freitas Dueling Network Architectures for Deep Reinforcement Learning