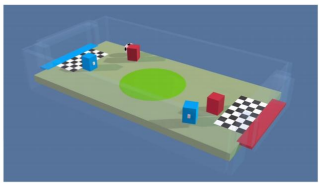
Cooperative-Competitive Multi-Agent Learning in Soccer Environments

Christopher Covert, Cameron McMillan, Patipan Pipatpinyopong
[cwcovert, cmac12, pipat001]@stanford.edu



ABSTRACT

Inspired by multi-agent cooperation, we are investigating the use of multi-agent deep reinforcement learning networks to play simple cooperative games. This project utilizes a simulated soccer environment developed by Unity to test a suite of deep reinforcement learning algorithms against each other in a simple game benchmark. In the 2 vs 2 player format, where each team is equipped with a striker (offensive agent) and goalie (defensive agent), we explore how agents can learn to be simultaneously collaborative and competitive. To avoid the influence of consecutive examples' strong correlations and interdependence, a randomized experience replay scheme was used to train the network. We explored the training result between Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO), each with small variations during training.

RESULTS

Table 1 shows the results on 500-run tests.

	Win (W) and Not Lose (NL) % Against:						
Model	Random	DQN_{base}	DQN_1	DQN_1^{mod}	DQN_2	DQN_2^{mod}	PPO_{base}
DQN_{base}	W: 13% NL: 81%	W: 0% NL:100%	W:18 % NL: 32%	W: 14% NL: 41%	W: 5% NL: 38%	W: 10% NL: 59%	W: 21% NL: 55%
DQN_1	W: 66% NL: 70%	W: 68% NL: 82%	W: 31% NL: 62%	W: 23 % NL: 49%	W: 28% NL: 48%	W: 37% NL: 57%	W: 63% NL: 68%
DQN_1^{mod}	W: 79% NL: 85%	W: 59% NL: 86%	W: 51% NL: 77%	W: 40% NL: 63%	W: 41% NL: 63%	W: 45% NL: 75%	W: 77% NL: 81%
DQN_2	W: 77% NL: 81%	W: 62% NL: 95%	W: 52% NL: 72%	W: 37% NL: 59%	W: 41% NL: 62%	W: 48% NL: 76%	W: 70% NL: 77%
DQN_2^{mod}	W: 53% NL: 74%	W: 41% NL: 90%	W: 43% NL: 63%	W: 25% NL: 55%	W: 24% NL: 52%	W: 39% NL: 59%	W: 49% NL: 71%
PPO_{base}	W: 53% NL: 70%	W: 45% NL: 79%	W: 32% NL: 37%	W: 19% NL: 23%	W: 23% NL: 30%	W: 29% NL: 51%	W: 46% NL: 58%

Table 1: Evaluation Results for the 2x2 Soccer Environment

DATASET

The dataset for training this model utilizes an experience replay method, meaning that the input set is generated at the time of training and added to a queue. For our application, the replay is a state-action set that Is randomly selected from the queue to decouple the influence of consecutive runs.

FEATURES

The striker has 6 possible actions: forward, backward, left, right, clockwise rotation, and counterclockwise rotation. The keeper has only four possible actions: forward, backward, left, and right. As for the state space, each agent obtains a 336 element observation vector, that corresponds to 7 rays within the 180 degree view from the front of the agent, with 6 sets of rays stacked in the vertical direction. Each ray consists of 8 values: binary classification of seven possible object types and the object's distance from the agent.

MODELS

A Proximal Policy Optimization (PPO) model was used to provide a baseline for a series of reward-modified Deep Q-Network (DQN) tests. Both models utilize the basic reward structure of Table 2.

Proximal Policy Optimization (PPO)

The PPO baseline was implemented with the following loss function:

$$L_t(\theta) = r_t(\theta)\hat{A}_t - \beta\,\mathrm{KL}[\pi_{\theta_{\mathrm{old}}}, \pi_\theta]$$

Deep Q-Network (DQN)

The DQN models were implemented with the following target assignment:

$$y_j = \begin{cases} r_j & \text{if episode terminates} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right) & \end{cases}$$

DQN_1, the striker receives the original reward outlined earlier where scoring a goal rewards +1 and -0.1 penalty, corresponding to a "neutral" striker that focuses on scoring and may play some defense.

DQN_2 corresponds to an "aggressive" striker that receives reward of +1 for scoring and 0 penalty if the team gets scored on.

DQN_3 corresponds to a "defensive" striker that gets reward of +1 for scoring and penalty of -0.6 if it gets scored on.

The *mod* tag represents models that have no penalty for not observing the ball.

Agent	Event		
	Ball Enters Opponent's Goal	Ball Enters Own Team's Goal	Existing
Striker	+1	-0.1	-0.001
Goalie	+0.1	-1	+0.001

Table 2: Basic Agent Reward Structure

DISCUSSION

The PPO model initially performed better against random and trained opponents.. Intermediate rewards were then introduced to help with training DQN models and made a significant improvement. One interesting result is that the penalty for the striker not seeing the ball doesn't seem to have much effect on the model's performance. We were able to see that the more aggressive strikers performed better in this game overall. We also found that introducing intermediate reward actually hurt the performance when training PPO models.

FUTURE

Future work includes further fine tuning of the reward structure. We would also like to expand the problem to include more agents, less environmental boundaries, more complex agent roles, and have different models trained against each other. Lastly, there are other deep reinforcement learning methods to be explored such as double DQN, dueling DQN, and imitation learning.

REFERENCES

[1] Condo, Ronny, José Ramón Lista, and Carlos Torre-Ferrero. "Time-varying formation controllers for unmanned aerial vehicles using deep reinforcement learning." arXiv preprint arXiv:1706.01384 (2017). https://arxiv.org/ftp/arxiv/papers/1706/1706.01384.pdf

[2] Gupta, Jayesh K., Maxim Egorov, and Mykel Kochenderfer. "Cooperative multi-agent control using deep reinforcement learning." International Conference on Autonomous Agents and Multiagent Systems. Springer, Cham, 2017. http://ala2017.it.nuigalway.ie/papers/ALA2017_Gupta.pdf

[3] Egorov, Maxim. "Multi-agent deep reinforcement learning." (2016). https://cs231n.stanford.edu/reports/2016/pdfs/121_Report.pdf

[4] Palmer, Gregory, et al. "Lenient multi-agent deep reinforcement learning." Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. International Foundation for Autonomous Agents and Multiagent Systems, 2018. https://dl.acm.org/citation.cfm?id=3237459

[5] Stanford Intelligent Systems Laboratory (SISL). GitHub Repository. https://github.com/sisl/MADRL

[6] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., and Petersen, S. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529. https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf

[7] Van Hasselt, H., Guez, A., and Silver, D. (2016, March). Deep reinforcement learning with double q-learning. In Thirtieth AAAI Conference on Artificial Intelligence. https://arxiv.org/pdf/1509.06461.pdf

[8] Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., and De Freitas, N. (2015). Dueling network architectures for deep reinforcement learning. arXiv preprint arXiv:1511.06581. https://arxiv.org/pdf/1511.06581.pdf

[9] Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized experience replay. arXiv preprint arXiv:1511.05952. https://arxiv.org/pdf/1511.05952.pdf

[10] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347. https://arxiv.org/abs/1707.06347

[11] Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P. (2015, July). Trust Region Policy Optimization. In Icml (Vol. 37, pp. 1889-1897). http://proceedings.mlr.press/v37/schulman15.pdf

[12] Juliani, A., Berges, V., Vckay, E., Gao, Y., Henry, H., Mattar, M., Lange, D. (2018). Unity: A General Platform for Intelligent Agents. arXiv preprint https://arxiv.org/pdf/1809.02627.pdf. https://github.com/Unity-Technologies/ml-agents

[13] Marcello Borges GitHub Repository. https://github.com/marcelloborges/Soccer-PPO