Imitation Learning for StarCraft II

Roman Ring
University of Tartu
roman.ring@ut.ee

Abstract

Imitation Learning (IL) is a subfield of Artificial Intelligence which deals with an agent learning to perform some task by observing expert demonstrations. In this project we explore the IL approach by training an agent in the StarCraft II environment. We show that the agent is able to achieve close to state-of-the-art results with reasonable amount of expert samples, requiring a fraction of the time it would take for an Reinforcement Learning based agent.

1 Introduction

A typical problem artificial intelligence attempts to solve involves an agent navigating in an environment while maximizing the resulting rewards. Researchers have explored many different ways to approach this problem – from rule-based agents that operated based on hand-crafted features provided by the domain experts; to agents independently learning desired behavior with artificial neural networks.

While the end goal of artificial intelligence based algorithms is typically in some real-world application such as robotics or finance, a simpler environment is necessary during development and evaluation stages to ensure that experiments are quick and easy to execute and replicate. Historically AI algorithms are benchmarked on various games as they have clear rules for navigation and rewards. With modern AI approaches, both classical board games [Silver et al., 2016] and modern video games [Mnih et al., 2015] are explored.

An intuitive approach to training an AI based agent the desired behavior is Imitation Learning (IL), where an agent is trained through demonstrations of a domain expert. This approach shares many drawbacks with supervised learning based approaches, namely it is difficult to achieve good generalization capabilities while keeping the training dataset size at a reasonable level. Despite its drawbacks, this approach is favorable to many researchers as there are many fields that would benefit from AI automation with human experts providing the demonstrations, e.g. self-driving cars.

In this project we explore imitation learning based approach to the problem of navigating a set of complex tasks in a modern video game environment – StarCraft II video game. We compare results of the implemented IL agent with baseline reinforcement learning (RL) based approaches. We then investigate possibility of a hybrid approach of pre-training an agent model with an IL agent and improving it with an RL agent.

2 Background

The environments described in this work are typically modeled through Markov Decision Processes (MDP), which are defined by a set of states S, a set of actions A, transition probability $P(s'|s,a)$ and immediate reward function $R(s'|s,a)$. The end goal for the agent learning to navigate in a given environment is an optimal policy π, which maps the best possible actions for any encountered state (determined by expected total rewards). A policy can be either deterministic or stochastic, discrete or continuous.

With imitation learning based approaches target policy π is learned by observing expert demonstrations, typically in an offline fashion. Demonstrations are usually provided beforehand by experts of the domain an agent will be navigating in.

An ideal outcome of imitation learning would be an agent that is able to not only replicate expert behavior, but also generalize well to previously unseen states. In practice this is typically unrealistic, as even minor deviation from optimal policy can lead to wildly different behavior over significant amount of time.

A popular (although naive) approach to imitation learning is behavior cloning, where an agent "memorizes" expert behavior by training on a dataset of (state, action) pairs. If action space is discrete then the problem of behavior cloning reduces to classification between predicted policy and expert actions in a supervised learning fashion. Agents model is typically trained with stochastic gradient descend (SGD) or its variants.

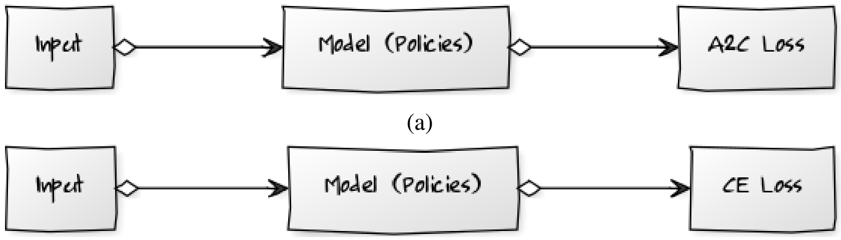
If the problem is reduced to classification then the underlying optimization target for the SGD can be defined with categorical cross-entropy loss:

$$J(\theta) = \sum_{i=1}^{M} y_i \log(\pi(a_i|s_i;\theta))$$

where M is the number of (a_i, s_i) samples of expert (state, action) pairs, y_i is one-hot encoded vector of the expert action a_i for i-th sample and $\pi(a_i|s_i;\theta)$ is the probability of the agent choosing action a_i given state s_i.

3 Methods

We implement the Imitation Learning agent by adapting the codebase of [Ring, 2018]. In particular, we fully reuse the model definition of the Advantage Actor-Critic (A2C) Agent and implement the necessary cross-entropy loss function in a custom IL Agent class (Figure 1). This was possible as A2C Agent was implemented in such a way that everything related to the agent model, including output policies, was defined separately from the agent class, passed to it in the constructor as a lambda function.



(b)

Figure 1: (a) A2C Agent pipeline with A2C loss at the end. (b) IL Agent pipeline, reusing A2C input and model layers, only replacing A2C loss with CE loss.

As the codebase is implemented in TensorFlow [Abadi et al., 2016], significant amount of time was invested into ensuring that learned model weights can be transferred between A2C and IL agents. In particular, when restoring a model the TensorFlow checkpoints system requires that all computational graph variables match. This resulted in several “hacks” whose only purpose was to ensure that computational graphs match between the two agents. For instance, we resorted to defining a fake value loss term (with zero weight) even though we never use the value variable in the IL agent.

While expert demonstrations in Imitation Learning typically refer to samples gathered by observing a human expert, this was difficult to achieve as part of this project as the StarCraft II game replay processing in PySC2 library was faulty at the time of writing. For this reason the expert data is gathered by launching reference A2C RL agent (trained to baseline scores) in inference mode and recording its state, action pairs as it plays the mini-games.

An action in the StarCraft II environment consists of multiple parts, e.g. a mouse-click action identifier and pixel coordinates of the screen or minimap for the click. Although the correct way to represent these actions would be through conditional joint probability, it would be computationally infeasible. In practice we make the simplifying assumption that sub-actions are independently distributed. In the context of IL Agent this means that SGD loss function is summed across all policy outputs:

$$J(\theta) = \sum_{i=1}^{M} \sum_{j=1}^{P} y_{ij} \log(\pi_j(a_{ij}|s_i;\theta))$$

where P is the number of policy outputs (in our case it is 14), $\pi_j(a|s)$ is the j-th policy of the agent, and y_{ij}, a_{ij} are expert actions for i-th sample and j-th sub-action.

Implemented agent is trained with gathered data in a typical supervised learning fashion by sampling random batches of sizes 128 to 512 (depending on the minigame) from the expert dataset and training with Adam optimizer [Kingma and Ba, 2014] with $1e-4$ learning rate. All model weights are initialized with He initializer [He et al., 2015]. Hyperparameter choices were made empirically by trying out from a set of three-four educated guesses and personal preferences.

4 Results

We have collected results on seven StarCraft II mini-games released with [Vinyals et al., 2017]. These mini-games contain small predetermined MDPs that test agents ability to solve a wide range of tasks, e.g. navigating to a target location, defeating opponent army, collecting resources.

To collect the results we loaded the trained model into the RL agent and executed it in inference mode (no training) on 32 environments, recording mean and standard deviation of the cumulative score for an episode. We compare these results with the baseline A2C RL agent, DeepMind FullyConv agent and human expert results. Please refer to Table 1 below for an overview.

Map Name	**Imitation Agent**	**A2C Agent**	**DeepMind**	**Human**
MoveToBeacon	23.6 ± 1.8	26.3 ± 0.5	26	28
CollectMineralShards	82.7 ± 12.3	106 ± 4.3	103	177
DefeatRoaches	136 ± 47	147 ± 38.7	100	215
DefeatBanelingsAndZerglings	192 ± 112	230 ± 106.4	62	727
FindAndDefeatZerglings	27 ± 2.23	43 ± 5	45	61
CollectMineralsAndGas	3250 ± 240	3340 ± 185	3978	7566
BuildMarines	0	0.55 ± 0.25	3	133

Table 1: Mean and standart deviation of total reward for an episode of the implemented agent relative to benchmarks.

The results nearly match reference A2C RL agent on all mini-games. The `BuildMarines` map with zero score is an environment that requires long-term economic planning and was unobtainable even for the reference A2C RL agent, which can only learn a policy that sometimes builds a couple of units most likely through sheer luck.

The Imitation Learning agent has struggled with replicating expert policy on the `FindAndDefeatZerglings` mini-game, most likely due to the partial observability of the mini-game. This situation provided an opportunity to test whether the RL agent is able to continue its learning process from the IL agent model weights.

We have trained the RL agent until it was clear that the average score has improved (training the agent back to reference results would take a significant amount of samples and thus skipped due to time considerations).

We have additionally investigated how the expert dataset size affects IL agents ability to learn (Table 2). We have concluded that the IL agent was able to reach target scores with 100,000 samples per task, though for some of the simpler tasks as low as 1000 samples was enough. For a human expert to gather 100,000 samples it would take approximately one to two hours of gameplay, which while tedious is not unrealistic.

Map Name / Samples	**1,000**	**10,000**	**100,000**
MoveToBeacon	12.2	23.6	23.6
CollectMineralShards	24	73	82.7
DefeatRoaches	10	45	136
DefeatBanelingsAndZerglings	34.2	53.9	192
FindAndDefeatZerglings	11	19	27
CollectMineralsAndGas	3250	3250	3250
BuildMarines	0	0	0

Table 2: Imitation Agent results for various number of samples provided during supervised training stage.

While these results are positive, they are quite unexpected given the complexity of the tasks, the sizes of the state and action spaces, and general instability of imitation learning based approaches. One possible reason for this phenomenon is that while action space is big and difficult to explore stochastically via RL based approaches, once it is learned the policy distribution entropy is quite low, meaning that majority of the probability mass is concentrated in a small subset of the action space which is easy to "memorize". In fact on some of the maps such as `DefeatZerglingsAndBanelings` or `CollectMineralsAndGas` moderate results can be achieved with just a few initial lucky actions.

Furthermore, these figures can be deceiving as the current target scores measured are significantly below those of an human expert. Additionally, the SC2LE mini-games test only a small portion of the tasks that a full StarCraft II game encompasses. Still, these proof-of-concept results show that Imitation Learning can be a viable approach to StarCraft II environment and provide clear path for future experiments with real human expert demonstrations.

5 Conclusion

In this work we have investigated application of Imitation Learning to the game of StarCraft II. We have successfully shown that an agent can learn good policies from expert demonstrations with reasonable amount of samples. We have further verified that an RL agent is capable of continuing the learning process after pre-training model weights with Imitation Learning. Implemented agent is available online `https://github.com/inoryy/pysc2-rl-agent` on the imitation branch.

In the future pre-training of model weights for subsequent RL agent training could be further explored by learning from human expert demonstrations on some of the harder tasks such as `BuildMarines`, where the current model fails to learn anything reasonable. Additionally, more sophisticated Imitation Learning approaches can be explored, such as DAgger [Ross et al., 2010] or GAIL [Ho and Ermon, 2016].

References

[Abadi et al., 2016] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. *ArXiv e-prints*.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE.

[Ho and Ermon, 2016] Ho, J. and Ermon, S. (2016). Generative Adversarial Imitation Learning. *ArXiv e-prints*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *ArXiv e-prints*.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

[Ring, 2018] Ring, R. (2018). Replicating DeepMind StarCraft II Reinforcement Learning Benchmark with Actor-Critic Methods.

[Ross et al., 2010] Ross, S., Gordon, G. J., and Bagnell, J. A. (2010). A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. *ArXiv e-prints*.

[Silver et al., 2016] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

[Vinyals et al., 2017] Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Sasha Vezhnevets, A., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., Quan, J., Gaffney, S., Petersen, S., Simonyan, K., Schaul, T., van Hasselt, H., Silver, D., Lillicrap, T., Calderone, K., Keet, P., Brunasso, A., Lawrence, D., Ekermo, A., Repp, J., and Tsing, R. (2017). StarCraft II: A New Challenge for Reinforcement Learning. *ArXiv e-prints*.