On the Generalization of Indoor Navigation Agent

Hao Sun

December 2018

1 Challenge of Indoor Navigation

Current navigation tasks often take first-person views as inputs and predict a sequence of actions toward goals in the environment[6]. This process usually consists of two modules: Mapping and Planning. The former converts the views from first person perspective into egocentric maps, while the latter uses these maps alongside goals to predict navigational actions. The mapping procedure belongs to the field of computer vision and is relatively well solved[4][13][22], while the planning procedure has not been solved with satisfactory yet.

In practical scenarios, a navigation agent is asked to give action advice based on the surrounding environment and target location, e.g., a search and rescue robot in a conflagration needs to design a safe rescue route according to its surroundings and the location of people awaiting rescue. Navigation agents at these scenarios need to make decisions quickly and accurately in previously unseen, sometimes partially observable environments, which requires the navigation algorithm to be accurate, robust and universal.

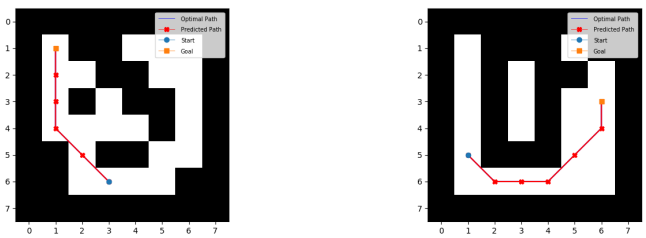
Reinforcement learning (RL) is promising in such setting but an important problem need to be solved: the "Over-fitting" problems arose here, i.e., the trade-off between accuracy and generality of a policy $\boldsymbol{\pi}$.

2 Generalization Problem of Reinforcement Learning Agents

In most situations in the setting of RL, people care about the dilemma of exploration and exploitation, i.e., should the agent choose an optimal action to get higher reward or choose a non-optimal action to explore more about the environment? Under normal cases, another fundamental dilemma of generalization and exploitation is always neglected.

The most famous RL algorithms at present, including DQN[20], A3C[11], TRPO[16], PPO[17] etc., consider the improvement of performance on Atari game environments or on Gym[3]. Their evaluation metric is always the average reward growth curve during the training process. Under these circumstances, the performance of a given algorithm actually represents the learning ability as well as the efficiency of learning process of the algorithm.

The success those algorithms demonstrates the potential of RL to approach intelligence. Nevertheless, genuine intelligence is more than familiarizing a agent with a particular environment. Take the environment of GridWorld for example (Figure 1), the task of an agent in such environment is to find a way to the goal point (the orange point in Figure 1) from a given start point (the blue point in Figure 1) as soon as possible. Given a static map as input, it's not hard for a traditional RL algorithm like Q-Learning[21] in tabular settings to find a shortest path between the start point and the goal point. However, when it comes to generalization, which requires agents to make decisions in a new map like Figure 1(b) after trained in the environment of Figure 1(a), the problem gets thorny.



(a) map1 (b) map2

Figure 1: The GridWorld Environment[19], where the black blocks are obstacles and white blocks are feasible regions

This problem can be formalized as follows: we want to train an agent based on limited number of environments, will the agent be able to solve similar problems in any environment of the same type? To solve such problem, the agent must be able to learn to "plan", i.e., able to generalize in a group of environments in stead of in a certain environment on the one hand, on the other hand, the agent must also be able to perform well in each environment during training process to gain more reward, i.e., able to exploit every single environment to its limits, which forces the agent to learn about the information that can promote its skills in reward-taking. Just as greedy algorithms sometimes fail to converge to the global optimal solution, the attention paid to solve single environment problem here may draw the policy away from universal optimal solution, i.e., the generalized planning ability. In general, RL agents are trained step by step and case by case, hence the only type of method we can utilize is learn as much as possible from each state-action-reward pair, which is exactly a greedy approach.

3 Traditional Algorithms

3.1 Dijkstra

3.2 A*

4 Related Work

4.1 Value Iteration Networks

Value Iteration Networks (VIN)[19] is proposed to tackle the above problem. The agent in VIN is trained using imitation learning (IL), in which tens of thousands of maps and the corresponding paths are given, thus the problem is transformed into supervised learning problem. The author proposed a novel network structure to avoid over-fitting. Although VIN is able to be trained with RL, the performance in such scenario is not guaranteed. The IL approach limits generality and scalability of VIN.

4.2 Graph Convolution Method

Generalized Value Iteration Network (GVIN)[12] is proposed as an improvement of VIN. GVIN take advantage of Graph Convolution (GC) and episodic Q-learning and outperforms VIN in terms of success rate. But GVIN needs to transform 2D maps into graphic model and the corresponding convolution kernels need to be designed carefully, thus the generality and flexibility of GVIN is limited.

4.3 DeepMind Lab

DeepMind Lab (DMLab)[2] is a first-person 3D game platform. It is tailored for machine leaning and enables multiple tasks. Some of its levels are relevant to the indoor navigation task: 1. Fruit gathering levels require the agent to do positive goal oriented gathering while avoiding the negative objects; 2. Navigation levels with static map test the agent's ability to find their way to a goal in a fixed maze remains the same across episodes; 3. Navigation levels with random generated mazes test the agent's ability to explore and exploit its knowledge in limited time durations.

4.4 Auxiliary Tasks

Mitoedki et al.[10] proposed that auxiliary tasks are useful for the learning process of RL agents. Specifically, learning to predict the depth of first-person views as well as predict loop closure classification provide significant improvement. Besides the tricks mentioned above, this work used stacked LSTM to prevent policy saturation.

4.5 Learning Over Subgoals

Stein et al. [18] proposed that subgoals cam help agents to navigate in structured unknown environments. In their approach, classical planning techniques are introduced to navigate toward subgoals, thus the arrival of the final goal is guaranteed. Their training data consists of hundred maps and the corresponding data from each run of optimistic planner.

4.6 Hindsight Experience Replay

Hindsight Experience Replay[1] proposed that for multi-goal tasks, experience replay can be improved to be more sample-efficient by learning through failure. Their motivating example of bit-flipping is in fact a navigation

environment. More concretely, finding the goal state in a bit-flipping problem with N bits is exactly the same as finding a path along edges of a N-dimensional unit cube.

5 Promising Approaches

5.1 Auto-encoder Pre-processing

Several previous work tried to combile Auto-encoders (AEs) with reinforcement learning as pre-processing[7][9][5]. AEs are used to reduce the number of training trials[7] or learn state representations[9][5].

In the task of indoor navigation, we may apply Auto-encoder Pre-processing (AEP) to improve generality of agents. The intuition of AEP net is that if our training set and test set are i.i.d, the emergence of over-fitting can be reduced, therefore, the generality of model will be improved.

AEP uses AutoEncoders with BatchNormalization to pre-train both the training data set and the test data set simultaneously, minimizing the MSE of input images and output images. And then we may use the encoded layer, with the number of neurons to be determined as needed in various tasks, to act as input layer of later networks. I tried to use AEP in both unsupervised star-galaxy classification task and prostrate MRI segmentation task and both result in significant performance promotion.

5.2 Imitation Experience Replay

Practical limitations associated with delayed rewards[8] make it difficult for navigation agents to learn policies that can effectively navigate large-scale environment[18]. Imitation learning solves the problem of delayed rewards by training with supervised learning, thus combining imitation learning with RL provides a possible approach.

Experience replay (ER) is used in off-policy RL to get higher data efficiency and better convergence behavior, removing correlations between the experience gained and learned during the training process[20]. Later, PER[15], HER[1] were proposed as improvements.

HER pointed that RL agents can learn not only from success but also from failure by universal value function approximators (UVFA)[14]. We can improve HER in an imitation learning approach in indoor navigation tasks, namely Imitation Experience Replay (IER). As there are multiple goals in the indoor navigation tasks, IL can be used together with UVFA to predict actions leading to pseudo-goals, i.e., any latter state in a state sequence. Moreover, the above process can be armed with any RL algorithm, promoting data efficiency and stability.

Figure 2 shows preliminary results of IER. The blue lines are VIN trained by DQN while the orange and green lines are trained by DQN with IER. The IER bring about significant improvement in terms of test roll out accuracy.

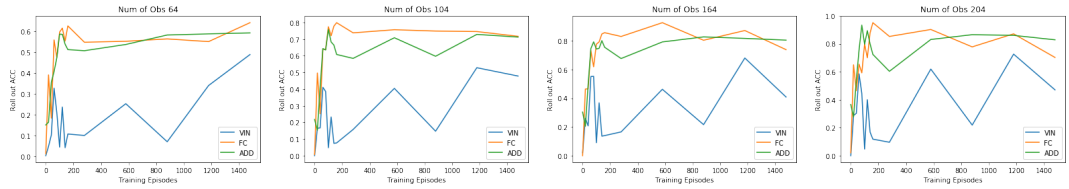


Figure 2: IER experiment results in environments with different number of obstacles, FC and ADD represents two combination method of IER and DQN

References

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob Mcgrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. 2017.

[2] Charles Beattie, Joel Z. Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, Julian Schrittwieser, Keith Anderson, Sarah York, Max Cant, Adam Cain, Adrian Bolton, Stephen Gaffney, Helen King, Demis Hassabis, Shane Legg, and Stig Petersen. Deepmind lab. *CoRR*, abs/1612.03801, 2016.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[4] Lin Chien-Chuan and Wang Ming-Shi. A vision based top-view transformation model for a vehicle parking assistant. *Sensors*, 12(4):4431–4446, 2012.

[5] Karol Gregor and Frederic Besse. Temporal difference variational auto-encoder. 2018.

[6] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. pages 7272–7281, 2017.

[7] Daiki Kimura. Daqn: Deep auto-encoder and q-network. 2018.

[8] Jens Kober and Jan Peters. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[9] Sascha Lange and Martin Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *International Joint Conference on Neural Networks*, pages 1–8, 2010.

[10] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J. Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell. Learning to navigate in complex environments. *CoRR*, abs/1611.03673, 2016.

[11] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. 2016.

[12] Sufeng Niu, Siheng Chen, Hanyu Guo, Colin Targonski, Melissa C. Smith, and Jelena Kovačević. Generalized value iteration networks: Life beyond lattices. 2017.

[13] Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C. Berg. Transformation-grounded image generation network for novel 3d view synthesis. 2017.

[14] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 518 of *PMLR*, pages 1312–1320, 07–09 Jul 2015.

[15] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *CoRR*, abs/1511.05952, 2015.

[16] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *Computer Science*, pages 1889–1897, 2015.

[17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. 2017.

[18] Gregory J Stein, Christopher Bradley, and Nicholas Roy. Learning over subgoals for efficient navigation of structured, unknown environments. In *Conference on Robot Learning*, pages 213–222, 2018.

[19] Aviv Tamar, Sergey Levine, and Pieter Abbeel. Value iteration networks. 2016.

[20] Mnih Volodymyr, Kavukcuoglu Koray, Silver David, Rusu Andrei A, Veness Joel, Bellemare Marc G, Graves Alex, Riedmiller Martin, Fidjeland Andreas K, and Ostrovski Georg. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[21] Christopher J. C. H. Watkins and Peter Dayan. Q -learning. *Machine Learning*, 8(3-4):279–292, 1992.

[22] Xinge Zhu, Zhichao Yin, Jianping Shi, Hongsheng Li, and Dahua Lin. Generative adversarial frontal view to bird view synthesis. 2018.