Knowledge-free domain-independent planning for games

Raphael Lopes Baldi
Pontifical Catholic University of Rio Grande do Sul, Brazil
Ipiranga Av., 6681, 90619-900
Porto Alegre, RS, Brasil

Abstract

Classical planning algorithms require us to first formalize problems as a symbolic model. That represents a knowledge acquisition bottleneck since it requires us to either get help from a specialist or acquire the knowledge ourselves, after which we need to describe the problem's domain formally. In both situations, the person in charge of creating the model needs to collect all states and transitions to use a classical planner. In this work, we intend to explore a technique to learn a transition function from the latent representation of the domain's states, and use it as input for an A* based planner.

Introduction

Planning is one of the various techniques Artificial Intelligence researchers use to solve problems. It is a process to select and organize actions to get to a target environment's state, given the current environment's state. Agents are capable of, using some formalization of the problem's domain, anticipate the outcome of actions while creating the plan (ordered sequence of actions) to achieve the goal state. Automated planning is the area of AI in which we design, implement and evaluate the deliberation process to select and order actions, computationally [2].

One of the most time-consuming tasks when dealing with automated planning is the creation of a formal model for environments. Usually, the job requires a broad knowledge of the system and this knowledge not always falls under the AI developer's expertise, which makes the modeling task time-consuming and prone to errors [1]. The primary objective of this work is to use a technique to automatic extract the domain definition from images (game's frames), and use that definition with a planner to make the agent capable of playing Atari games.

It is hard to define games as a planning problem manually due to the exponential explosion of states. Taking the Atari video game as an example, the console renders the screen 60 times (or frames) per second, and for each of those frames, the player can choose one action from the 17 available - including the no operation action. If we were to expand all possible transitions, at the end of the first second, we would have to evaluate 17^{60} transitions. In 2017, Asai and Fukunaga introduced Latplan, which purpose goes in line with our objective: it uses a series of neural networks to learn an encoding of the domain, and a transition function that allows a planner to use search methods - like A* - by sampling such networks.

Our goal is to develop an approach to automatically play video games using planning algorithms, without prior knowledge of the problem's domain, and without the need to engineer the domain formalization by using a technique for extracting the domain definition from game images and memory [1].

Technical approach

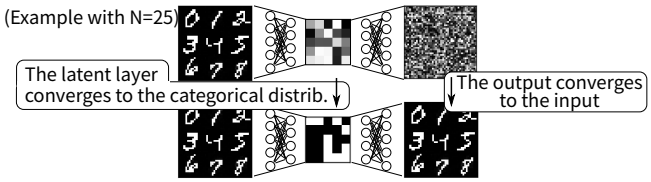

Figure 1: State autoencoder latent layer converges into a categorical distribution

Our first objective will be to create an autoencoder contain bottleneck layer, to extract a latent representation of e ate available. The idea behind it is that if the autoencoder can adequately reconstruct the input frames - representing the game's states -, then the bottleneck layer must hold enough information to represent such states. Our first attempt was to use a Gumbel-Softmax layer, and get the latent layer to converge into a categorical distribution (see Figure 1), as Asai and Fukunaga proposed in their work. Doing so would allow us to use a direct conversion between the latent representation of the state transitions to PDDL Figure 2), but we could not find a neural network architecture that would converge into a categorical distribution using Atari images.

The second network we have to build is a state discriminator: a neural network that will be trained to detect valid states from their latent representation. We will use it as a filter to prune out invalid states resulting from applying the transition function during the planning phase.

Analogously to the networks above, we need two others to deal with the system's actions. Initially, an action autoen-

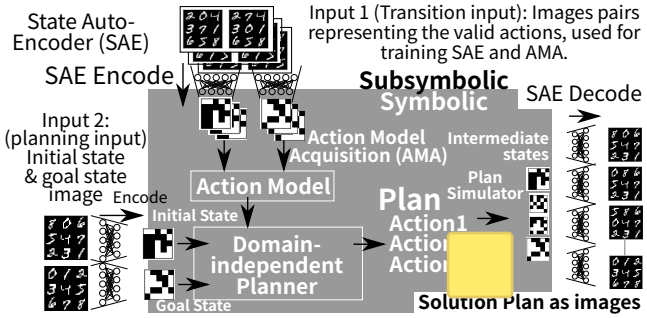

Figure 2: Categorical distribution as input for planning

coder (Figure 3), which learns a one-hot latent representation of the transitions in the system. This network serves to reduce the dimensionality of applicable actions which then allows for enumerating actions during the search. Finally, we also need to train an action discriminator; its objective is to identify transitions that are valid in the domain. The action discriminator is necessary since the action autoencoder can learn the number of actions and their effects, but not address their applicability.

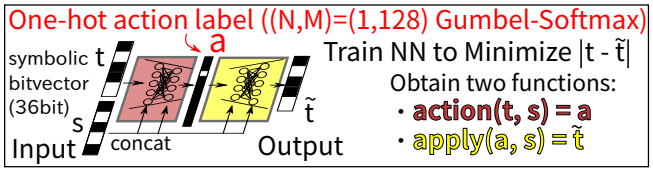

Figure 3: Action autoencoder

Ideally, we would use Asai and Fukunaga technique as is, since it relies on forcing the latent layer into a categorical distribution, using a reparametrization technique [3]. During our initial evaluation of the technique, we were not able to find a suitable neural network architecture for the state autoencoder to converge using a Gumbel Softmax distribution as the bottleneck layer. That means we need to adapt the other components in the system to accept non-binary inputs.

Project management

We have most of the architecture in place, meaning we have to run experiments to validate the technique in a domain more complex than the ones Asai and Fukunaga used for the article we are basing our research. The complete work will comprise the author's Final Undergraduate Work II, to be submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science, and as so is already in development.

Table 1 presents a tentative schedule for the tasks necessary for us to complete the work we are proposing:

1. Organize the infrastructure to accommodate Atari frames.
2. Train all the networks.
3. Integrate the networks into the planner.
4. Collect results and compare our approach to other planners on the same domain.
5. Write the final report.

Table 1: Tentative time schedule for the project

Activity	W1	W2	W3	W4	W5
1	X	X			
2		X	X		
3		X	X		
4		X	X	X	
5				X	X

Conclusion

This work proposes to use deep learning as a way to acquire knowledge from a given domain and use it as input for planning. If we manage to get the planner working in a complex domain, using only game frames as input, other possibilities might unfold, like using the architecture to allow robots to plan in environments where they are interacting with humans. We expect the planner to be able to find satisfiable plans for a short horizon, as even after encoding domain's transitions we still expect to see a state explosion.

References

[1] Masataro Asai and Alex Fukunaga. Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary. *CoRR*, abs/1705.00154, 2017.

[2] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning and Acting*. Cambridge University Press, New York, NY, USA, 1st edition, 2016. ISBN 1107037271, 9781107037274.

[3] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *Fifth International Conference on Learning Representations (ICLR)*, 2017.