Neural Planning Agents: Architecture

Pranav Vajreshwari
IMT2020544

GOAL

Reinforcement Learning involves conveying a goal through a reward function, which the agent implicitly achieves by maximizing a single scalar cumulative reward. The learnt policy is specific to each particular goal and there is no generalisable information even across very similar goals.

We humans have a generalist world model which we can navigate to achieve any goal in that domain. My work essentially is to implement this idea of general map learning + deterministic planning rather than task-specific reward maximisation.

Deterministic Planning algorithms work efficiently for smaller state spaces, however, for pixel-level observations from complex environments, deterministic algorithms fail. The solution is to learn a compact state space containing meaningful abstraction of the environment so that planning algorithms can be used on it.

Many existing models obtain latent state spaces (e.g. Curiosity-Based Methods) however, they capture the image appearance similarity between states and not semantic navigational distance, hence it is not possible to make a graph that can be used for planning.

The goal is to:

1) Build an autoencoder with a loss function that extracts reachability between states as latent distance.
2) Perform segmentation in the latent space
3) Build a navigational graph, then use a planning algorithm to navigate to the desired goal.

BACKGROUND

A. Model-Based RL and Model-Free RL

Model-free methods directly learn a policy from observations, while Model-Based methods learn an MDP from observations, and then plan over this learned model to act. Planning here essentially involves looking ahead for a fixed length and picking the action that gives the maximum n-step cumulative reward.

B. General Planning Methods

While lookahead n-step cumulative reward is a type of planning, the most general definition of planning involves finding a sequence of actions (a1,a2,..) to reach a goal state G from an initial state S. We will refer to these types as reward-based and general planning.

LITERATURE SURVEY AND GAP ANALYSIS

Predictive Representations in Hippocampal and Prefrontal Hierarchies (2022): Shows that we use learned representations of relational structures to explore and reach goals. These predictive models are also on a multi-scale hierarchy, and we plan on semantic levels to achieve a goal. This paper is the main inspiration for this research.

Hierarchical Reinforcement Learning: A Survey and Open Research Challenges (2022): Inspired by hierarchical abstractions in humans, HRL utilizes forms of temporal- and state abstractions to enable hierarchical reasoning. However, most of these methods rely on manually defined subgoals or tasks. Fully automatic discovery (the semi-automatic method of option-critic is the best we have so far) of meaningful hierarchies remains an open challenge. The proposed learned graph hirearchy is a potential solution.

The following papers address the problem of state space compression to enable planning:

- **AlphaGo: Using machine learning to master the ancient game of Go (2016):** This is a classic paper that combines RL and planning. The key approach is to first learn a representative value function and a policy through supervised learning from expert demonstrations. Then the RL agent plays against itself. It uses MCTS to pick the best action, where it searches for top-k best states and actions instead of the whole state space according to the expert value function.

  Gap: While this succeeds in performing planning, it is not practical to obtain expert data for each problem, and the environments aren't as discrete and deterministic as Go, therefore we look for better methods.

- **Learning Latent Dynamics for Planning from Pixels (PlaNet) (2018):** This paper learns a state encoder $f : s- > s'$ and a latent action conditioned predictive model $f : s, a- > s'$ from observations. During acting, the planner checks out different action trajectories (the reward-based planning mentioned earlier).

Gap: The state encoder derives representation considering only image characteristics and not semantic content, and reward engineering is still required.

- **Classical Planning in Deep Latent Space (2021):** Introduces a model called LatPlan that is a neurosymbolic planner. It works on puzzle-like problem domains 8-puzzle, Blocks World, etc. Here the images of possible state transitions are shown and the neural network infers the symbolic state from the images. Then, a classical PDDL solver is used to plan towards the goal state.

  Gap: While LatPlan does support general planning and does not need reward engineering, it does not apply to high-dimensional problems or general RL problem domains (like robotic control).

- **Planning to Explore via Self-Supervised World Models (2020):** Here, the agent learns a self-supervised latent state space of the world just like in PlaNet. However, unlike PlaNet which can know the novelty of a state (the loss signal for the autoencoder) only after visiting it, Plan2Explore uses a recursive rollout of future states using its current dynamics model (thus the word 'planning') and calculates n-step in the future loss signal. This helps the agent learn a better representation.

  Gap: While it speeds up learning of a latent representation, it still doesn't support general planning and needs reward engineering.

- **Value Memory Graph: A Graph-Structured World Model for Offline Reinforcement Learning (2022):** This Paper creates an alternative to Hirearchial RL, where there is a high-level policy and an action translator to convert it into a low-level policy. However, instead of manually defining a high-level policy or learning a neural network high-level policy, they obtain a VMG- an abstract MDP by segmenting state representations. This work uses reachability estimates between states, performs contrastive learning, learns state representations and then performs the segmentation.

  Gap: This work uses an action encoder and a state encoder trained on a contrastive loss together to get the reachability-defined latent state space. However, contrastive loss needs massive amounts of training data, and we look for better ways to find reachability estimates. Even though the work makes reachability-defined compressed MDP, it doesn't explore the possibility of using planning algorithms and focuses only on learning an RL policy using rewards. I explore methods to find reachability estimates more easily and come up with ways to perform general planning on a high-level graph of the environment instead of using reward engineering.

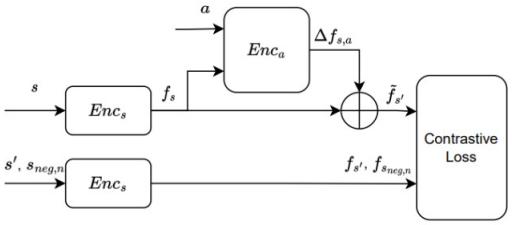
ARCHITECTURE

Learning reachability

I consider the offline RL setting throughout this work, accounting for the computational budget. In a contrastive learning approach to reachability, we pick $(s1, a, s2)$ and train:

- Encoder $f : s- > s', s'$ is the latent space
- Action conditioned state predictor $\Delta f : s, a- > s'$

The encoder output is $f(s) + f(s, a)$, which we force to be close to the known next state $f(s')$ and further away from all the other next states s_{neg}



$$L_c = D^2(\tilde{f_{s'}}, f_{s'}) + \frac{1}{N}\sum max(m - D^2(\tilde{f_{s'}}, f_{s_{neg,n}}), 0)$$

The states that occur close by are close together in this latent space.

Modification to support Planning

In a planning problem, the goal state is defined either explicitly in s or with a qualifier $\psi : s- > g$ which maps a state to a goal that the state satisfies. This function encodes the semantics of the goal space g. The earlier mentioned autoencoder doesn't consider the semantics of goal space which is essential if we need to build a planner on the latent space.

Model Architecture

The autoencoder would learn distance based on goal reachability if we have a function that can calculate the number of steps between the goals: $V(g1, g2)$

$$L(g1, g2) = D(F_E(g1), F_E(g2)) - V(g1, g2) \ldots Eq1$$

To estimate this function V, we follow the following steps:

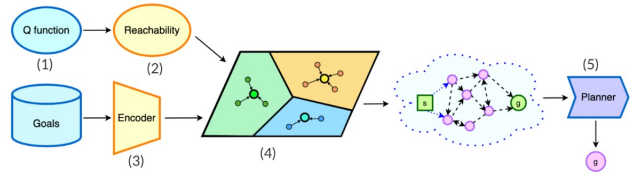
1) Inspired from the work in Goal conditioned RL, we define a special reward structure where the rewards are -1 for every action from s_t till the goal state $\psi(s_k) = g$ is reached and then the rewards are 0. By finding the optimal $q(s, a, g)$ value for s_0 under this setting, we are essentially calculating its distance $D(s, a, g)$ from the goal state.

$$Q(s,a,g) = \sum_{t=0}^{D(s,a,g)-1} \gamma^t \cdot (-1) + \sum_{t=D(s,a,g)}^{T-1} \gamma^t \cdot 0 = -\frac{1-\gamma^{D(s,a,g)}}{1-\gamma}$$

2) By using $D(s,a,g)$ as the label we can fit the function V (taken to be a simple neural network) for $\phi(s_t)$ and $\phi(s_k)$:

$$\min_V (D(s_t, a_t, \phi(s_k)) - V(\phi(s_{t+1}, \phi(s_k)))^2$$

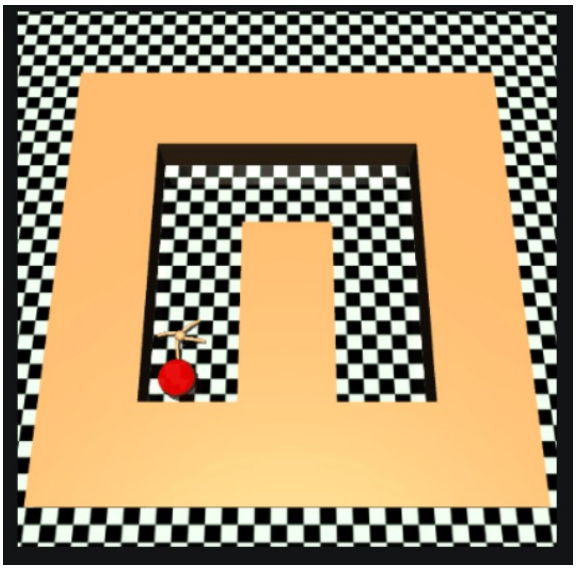
3) Now, after finding out the function V, we sample (s, s') transitions from our episode data and fit the autoencoder in $Eq1$
4) We then perform segmentation on this latent space
5) Use a planner to reach the final goal state gT.



(1) Multi goal q-learning -> (2) Estimating reachability V -> (3) Training state autoencoder -> (4) Segmentation in Latent space -> (5) Planner on the graph

EXPERIMENTS

I propose to run this architecture on tasks that require following landmarks on the way to the final goal.
A great application would be AntMaze environment by OpenAI Gym:



The hope is that the agent learns the landmarks at each corners in the maze and uses a planner to connect these points and finally navigates with this plan.

Later, we expand the testing to the D4RL (Fu et al., 2020) dataset on three domains: Kitchen, AntMaze, and Adorit.

All these tasks have large state and action spaces, and would be ideal to test if the model is being able to form graph-structured abstractions to navigate efficiently.

REFERENCES

[1] Predictive Representations in Hippocampal and Prefrontal Hierarchies (2022) by Iva K Brunec 1, Ida Momennejad 2
[2] Hierarchical reinforcement learning : a survey and open research challenges by Hutsebaut-Buysse, Matthias Mets, Kevin Latré, Steven
[3] Learning Latent Dynamics for Planning from Pixels by Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, James Davidson
[4] Classical Planning in Deep Latent Space by Masataro Asai, Hiroshi Kajino, Alex Fukunaga, Christian Muise
[5] Planning to Explore via Self-Supervised World Models by Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, Deepak Pathak
[6] Value Memory Graph: A Graph-Structured World Model for Offline Reinforcement Learning by Deyao Zhu, Li Erran Li, Mohamed Elhoseiny
[7] Plan2vec: Unsupervised representation learning by latent plans. In Learning for Dynamics and Control by Ge Yang, Amy Zhang, Ari Morcos, Joelle Pineau, Pieter Abbeel, and Roberto Calandra
[8] Sutton, R. S. Dyna, an integrated architecture for learning, planning, and reacting. ACM Sigart Bulletin, 2(4):160– 163, 1991.
[9] Srinivas, A., Jabri, A., Abbeel, P., Levine, S., and Finn, C. Universal planning networks. April 2018.
[10] D4RL: Datasets for Deep Data-Driven Reinforcement Learning by Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, Sergey Levine