Neural Planning Agents: Architecture

Pranav Vajreshwari
IMT2020544

Abstract—**Reinforcement Learning is based on the idea that agents need to 'reinforced' to get them to achieve a specific goal. However, humans and animals have generalist learned representations of the world, over which they can plan to reach any desired goal. Recently, LLMs have popularised this paradigm of learning a general representation with data that can be used in any required goal easily. In this work, I come up with a way to perform state abstraction of any general MDP environment using episodic state transition data, learn useful 'landmarks' in the environment, and use these landmarks to deterministically plan a path to a desired goal. This is the only work as far as I am aware that uses deterministic planning using 'learned' landmarks in a general MDP.**

Index Terms—**NeuroSymbolic methods, Deterministic Planning, Self-Supervised Learning, Markov Decision Process**

INTRODUCTION

Reinforcement Learning involves conveying a goal through a reward function, which the agent implicitly achieves by maximizing a single scalar cumulative reward. Value-based methods or Policy-Based methods learn the reward information conditioned on the state [Sutton and Barto, 2018]. Deep Learning methods parametrize value/policy function such that similiar states have similiar function values. These methods can sometimes achieve generalization across different goals or environments, but require fine-tuning and retraining on the new environments [Cobbe et al., 2019]. My method on the other hand adopts instantly to any new goals within its graph-structured world model without any retraining.

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 small state and action spaces, where the dynamics of the environment are known. This is because the algorithm can realistically visit all the states and search over the possible actions [Hart et al., 1968]. However, when we get to exponentially larger state spaces and continuous action spaces, the search space for the deterministic algorithms explodes combinatorily [Li et al., 2009]. My Proposed solution is to decompose these large state spaces into semantically rich, compact state spaces capturing the abstraction required for deterministic planners to work.

Many existing models obtain latent state spaces (e.g. Curiosity-Based Methods [Pathak et al., 2017]) using autoencoders which encode the raw state observations into compact, latent representations. However, they capture the abstraction of image appearance similarity between states for and not semantic navigational distance, hence it is not possible to make a graph that can be used for planning.

The goal here is to build an autoencoder that can take the raw state space and build a compact state space that captures reachability between states as latent distance. In this obtained state space, the high density points (which are much lesser than the number of raw states) are identified and assumed to be the semantically important landmarks. Using these points, we build a graph structured model of the world which is small enough that a deterministic planning algorithm can be used on top of it.

BACKGROUND

A. Model-Based RL and Model-Free RL

Model-free methods directly learn a policy from observations, while Model-Based methods (as defined in [Sutton and Barto, 2018]) 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. In some methods, the model can be known already, like the tree structure game spaces in the game of Go/Chess. Here too, methods like Monte-Carlo Tree Search lookahead for a fixed length to determine the best actions.

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.

C. Goal conditioned RL (GCRL)

Standard RL only requires the agent to finish one specific task defined by the reward function while GCRL [Liu et al., 2022] seeks to learn a policy depending on the various goals possible within an environment. GCRL hopes to learn a policy $\pi(a|s, g)$ where g is the current goal within the environment. By training on multiple goals, the hope is that the goal-parameterized policy learns generalizable information across goals.

Literature Survey and Gap Analysis

Predictive Representations in Hippocampal and Prefrontal Hierarchies [Brunec and Momennejad, 2022] Shows that humans 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 [Hutsebaut-Buysse et al., 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** [Silver et al., 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)** [Hafner et al., 2019] 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** [Asai et al., 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** [Sekar et al., 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** [Zhu et al., 2023] 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.

Problem Statement

Our goal is to:

1) Learn an autoencoder for the states where the latent space captures reachability.

2) Perform Segmentation to find cluster points in this space and
3) Use a Planning algorithm to reach any given goal where these learned cluster points acts as 'landmarks' to navigate to the final goal.

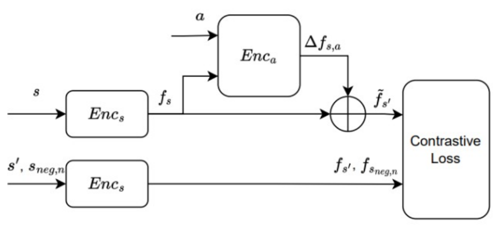
Let's consider an existing approach to learning reachability using contrastive loss given in [**?**].

Using agent's trajectories, the tuple $(s1, a, s2)$ is used to train:

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

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

Fig. 1.



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

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

Modification to support Planning

This approach still considers episodes generated by a policy aiming to maximize one single goal and hence, the learned latent space won't contain generalizable information across all possible goals within the environment.

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 all the possible goals in 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. Thus, we wish to find an alternate way to train the autoencoder that considers the goal space.

Once we have the right latent space, we wish to segment the learned state mappings based on density to select points that are most often visited and thus represent semantically important landmarks common across different goals.

Given a new goal, we can use any planning algorithm like Djikstra's or Floyd's algorithm to find the shortest route across these landmarks to reach the final goal.

ARCHITECTURE

This section gives a 5 step approach for the problem statement, summarized in Fig 2.

A. Learning reachability

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

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

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

Step 1: Inspired from the Goal conditioned RL work, we define a special reward structure where the rewards are -1 for every action from st till the goal state $\phi(sk) = g$ is reached and then the rewards are 0. By finding the optimal $q(s, a, g)$ value for s0 under this setting, we are essentially calculating its distance $D(s, a, g)$ from the goal state.

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

Here, Q learning is employed to maximise the objective:

$$J(\pi) = E_{g\sim p_g, r\sim\pi(g)}[\textstyle\sum_{t=0}^{T-1} \gamma^t \cdot R(s_t, a_t, s_{t+1}, g)]$$

Where p_g denotes the possible goal space, τ the trajectories taken using $\pi(g)$: the policy conditioned on the goal g.

Step 2: We sample (s, a, s') from our existing trajectories and calculate $D(s, a, g)$ using the obtained $Q(s, a, g)$. 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 $\phi(s_k)$:

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

Step 3: Now, we train the autoencoder given in Eq 1 using the same (s, a, s') samples and the $\phi(s)$ function (as an intermediate for the $V(\phi(s_t), \phi(s_k))$ function.

B. Segmentation of the Latent Space

The latent space captures the temporal distance between goals. By finding regions of high density in this space, we find our desired landmarks for navigation. We assume that these high-density points are peak local points in a Gaussian distribution. We also pre-decide the number of centroids N_c we wish to find and then find the best possible Guassian Mixture Model for the taken N_c.

Step 4: We learn the GMM by assuming a uniform prior for the c centroids and minimizing the Evidence Lower Bound (ELBO):

$$logp(z = f_E(g)) \geq E_{q(c|z)}[log(p(z|c))] - D_{KL}(q(c|z)||p(c))$$

We thus obtain the desired c's.

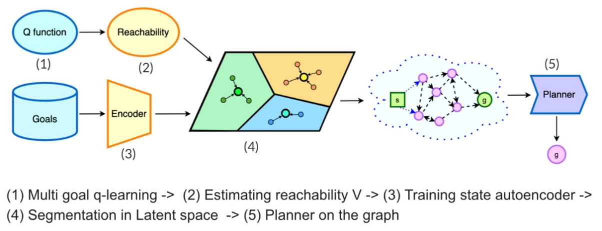
C. Planning on the Learned Landmarks

Having obtained the N_c centroids, for any given goal g the agent can navigate from its initial state by finding a path in the latent space that passes through these centroids. These centroids essentially act as subgoals that are identified by the agent and navigated through on the way to the final goal g. We will form a $N_c \times N_c$ Weight Matrix W, where W_{ij} gives the distance between centroids c_i and c_j as $V(c_i, c_j)$ (V is the distance metric learnt earlier). This Weight Matrix essentially defines a graph with the edge weights denoting the distance between them.

Step 5: Dijkstra's algorithm is applied directly on this graph W to output a path $(c_k, c_{k+1} \dots g)$ where g is our desired goal state (mapped to the nearest centroid c).

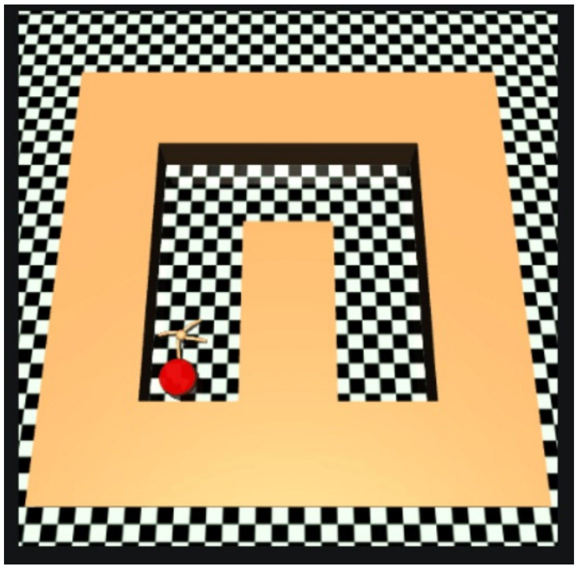
Each of the intermediate landmarks c_i acts as a subgoal for the agent, and the agent learns a policy $\pi(a|s, c_i)$ using the same GCRL framework mentioned earlier.

Fig. 2.


Q function
(1)
Reachability
(2)
Goals
Encoder
(3)
(4)
s
g
(5)
Planner
g
(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:



I focused only on the AntMaze environment due to time constraints. The hope was that the agent would learn the landmarks at each corner of the maze, and use a planner to connect these points and finally navigate with this plan.

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

While training our model each timestep took around 10s and the baseline paper showed an inflection above 0 reward only after a million timesteps, hence it was not possible for me to test the model with the time and computational constraints.

Conclusions and Future Work

I was able to make an algorithm that successfully performs abstraction of an environment independent of the reward function and contains generalizable information across all possible goals. Building upon GCRL, the main contribution was coming up with a distance metric that encodes semantic reachability among the states. The current segmentation method needs us to pre-specify the number of landmarks to be learnt, which can lead to redundant landmarks and unnecessary complexity for simpler cases. Building an algorithm to automatically identify the number of landmarks based on problem complexity can be an interesting future direction.

References

[Asai et al., 2021] Asai, M., Kajino, H., Fukunaga, A., and Muise, C. (2021). Classical planning in deep latent space. *CoRR*, abs/2107.00110.

[Brunec and Momennejad, 2022] Brunec, I. K. and Momennejad, I. (2022). Predictive representations in hippocampal and prefrontal hierarchies. *Journal of Neuroscience*, 42(2):299–312.

[Cobbe et al., 2019] Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. (2019). Quantifying generalization in reinforcement learning. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1282–1289. PMLR.

[Hafner et al., 2019] Hafner, D., Lillicrap, T. P., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. (2019). Learning latent dynamics for planning from pixels. In Chaudhuri, K. and Salakhutdinov, R., editors, *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 2555–2565. PMLR.

[Hart et al., 1968] Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

[Hutsebaut-Buysse et al., 2022] Hutsebaut-Buysse, M., Mets, K., and Latré, S. (2022). Hierarchical reinforcement learning: A survey and open research challenges. *Machine Learning and Knowledge Extraction*, 4(1):172–221.

[Li et al., 2009] Li, L., Wang, D.-Y., Shen, X., and Yang, M. (2009). A method for combinatorial explosion avoidance of ai planner and the application on test case generation. *2009 International Conference on Computational Intelligence and Software Engineering*, pages 1–4.

[Liu et al., 2022] Liu, M., Zhu, M., and Zhang, W. (2022). Goal-conditioned reinforcement learning: Problems and solutions. In Raedt, L. D., editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 5502–5511. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

[Pathak et al., 2017] Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 2778–2787. JMLR.org.

[Sekar et al., 2020] Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., and Pathak, D. (2020). Planning to explore via self-supervised world models. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 8583–8592. PMLR.

[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.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

[Zhu et al., 2023] Zhu, D., Li, L. E., and Elhoseiny, M. (2023). Value memory graph: A graph-structured world model for offline reinforcement learning. In *ICLR*. OpenReview.net.