World Model as a Graph: Learning Latent Landmarks for Planning Supplementary Materials

Lunjun Zhang [1 2] **Ge Yang** [3] **Bradly Stadie** [4]

1. Greedy Latent Sparsification

Algorithm 2 Greedy Latent Sparsification (GLS) for Latent Cluster Training

Given: Replay Buffer B, Encoder f_E.
Initialize: `LatentEmbeds` = {}. ▷ Set of embeddings selected.
Sample K achieved goals from B.
Sample $k \sim \{0 \cdots K-1\}$.
$\texttt{dist} = [\|f_E(g_1) - f_E(g_k)\|_2^2, \cdots, \|f_E(g_K) - f_E(g_k)\|_2^2]$
for i = 1 to M **do** ▷ Sub-sampling
  $k \leftarrow \arg\max \texttt{dist}[k]$
  Add $f_E(g_k)$ to `LatentEmbeds`.
  $\texttt{NEWdist} = [\|f_E(g_1) - f_E(g_k)\|_2^2, \cdots, \|f_E(g_K) - f_E(g_k)\|_2^2]$
  $\texttt{dist} = \texttt{ElementwiseMin}(\texttt{dist}, \texttt{NEWdist})$
end for
Optimize equation 5 on `LatentEmbeds`.

The Greedy Latent Sparsification (GLS) algorithm subsamples a large batch by sparsification. GLS first randomly selects a latent embedding from the batch, and then greedily chooses the next embedding that is furthest away from already selected embeddings. After collecting some *warm-up trajectories* before planning starts (see Table 1 below) during training, we first use GLS to initialize the latent centroids, and then continue to use it to sample the batches used to train the latent clusters. GLS is strongly inspired by (Arthur & Vassilvitskii, 2007), and this type of approach is known to improve clustering.

2. Graph Search with Soft Relaxations

In this paper, we employ a soft version of Floyd algorithm, which we find to empirically work well. Rather than simply using the $\min$ operation to do relaxation, the soft value iteration procedure uses a $soft\min$ operation when doing an update (note that, since we negated the distances to be negative in the weight matrix of the graph, the operations we use are actually max and softmax). The reason is that neural distances can be inconsistent and inaccurate at times, and using a soft operation makes the whole procedure more robust. More concretely, we repeat the following update on the weight matrix for S steps with temperature β:

$$w_{i,j} \leftarrow \sum_{k=1}^{N+1} \frac{\exp \frac{1}{\beta}(w_{i,k} + w_{k,j})}{\sum_{k'=1}^{N+1} \exp \frac{1}{\beta}(w_{i,k'} + w_{k',j})} \Big(w_{i,k} + w_{k,j}\Big) \tag{1}$$

Following the practice in (Eysenbach et al., 2019; Huang et al., 2019), we do the following initialization to the distance matrix: for entries smaller than the negative of d_max, we penalize the entry by adding $-\infty$ to it (in this paper, we use -10^6 as the $-\infty$ value). The essential idea is that we only trust a neural estimate when it is *local*, and we rely on graph search to solve for *global*, longer-horizon distances. The $-\infty$ penalty effectively masks out those entries with large negative values in the softmax operation above. If we replace softmax with a hard max, we recover the original update in Floyd algorithm; we can interpolate between a hard Floyd and a soft Floyd by tuning the temperature β.

3. Overall Training Procedure

Here we provide an overall training procedure for L^3P in **Algorithm 3**. Given an environment `env` and a training goal distribution $p(g)$, we initialize a replay buffer B and the following **trainble modules**: policy π, distance function D, value function V, encoder f_E and decoder f_D, latent centroids $\{\mathbf{c}_1 \cdots \mathbf{c}_N\}$.

Every K_{env} episodes of sampling, we take gradient steps for the above modules. The ratio between the number of environment steps and the number of gradient steps is a hyper-parameter.

4. Implementation Details

- We find that having a centralized replay for all parallel workers is significantly more sample efficient than having separate replays for each worker and simply averaging the gradients across workers.
- For Ant-Maze environment, we do grad norm clipping

[1]University of Toronto [2]Vector Institute [3]MIT [4]Toyota Technological Institute at Chicago. Correspondence to: Lunjun Zhang <lunjun@cs.toronto.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

Algorithm 3 Overall Training of L^3P

Given: Environment `env`, training goal distribution $p(g)$.
Initialize: Policy π, distance function D, value function V.
Initialize: Auto-encoder f_E and f_D, replay buffer $B = \{\}$.
Initialize: Initialize N latent centroids $\{\mathbf{c}_1 \cdots \mathbf{c}_N\}$

while *not converged* **do**
  for i = 1 to K_{env} **do**
    $g \sim p(g)$, $\tau \sim \pi(g)$ with L^3P planning
    add τ to replay B
  end for
  minimize equation 1 for Q parameterized by D in equation 3
  minimize equation 4 for V
  minimize $\mathcal{L}_{rec} + \lambda \cdot \mathcal{L}_{latent}$ for f_E and f_D
  minimize equation 5 for latent centroids
  update the policy: $\max_\pi \mathbb{E}_{s\sim B} Q(s, \pi(s, g), g)$
end while

by a value of 15.0 for all networks. For Fetch tasks, we normalize the inputs by running means and standard deviations per input dimensions.

- Since L^3P is able to decompose a long-horizon goal into many short-horizon goals, we shorten the range of future steps where we do hindsight relabelling; as a result, the agent can focus its optimization effort on more immediate goals. This corresponds to the hyper-parameter: hindsight relabelling range.
- During training, we collect 50% of the data without the planning module, and the other 50% of the data with planning. This corresponds to the hyper-parameter: probability of using search during train.
- At train time, to encourage exploration during planning, we temporarily add a small number of random landmarks from GLS (**Algorithm 2**) to the existing latent landmarks. A new set of random landmarks is selected for each episode before graph search starts (**Algorithm 1**). This corresponds to the hyper-parameter: random landmarks added during train.
- We find that collecting a certain number of *warm-up trajectories* for every worker before the planning procedure starts (during training) and before GLS (Algorithm 2) is used for initialization to help improve the planning results. This corresponds to the hyper-parameter: number of *warm-up trajectories*.

5. Hyper-parameters

The first table below lists the common hyper-parameters across all environments. The second table below lists the hyper-parameters that differ across the environments.

Parameter	Value
DDPG	
optimizer	Adam (Kingma & Ba, 2014)
number of hidden layers (all networks)	3
number of hidden units per layer	256
nonlinearity	ReLU
polyak for target network (τ)	0.995
target update interval	10
ratio between env vs optimization steps	2
Random action probability	0.2
Initial random trajs per worker	100
Hindsight relabelling ratio	0.85
Latent Landmarks & Auto-encoder	
number of hidden layers	2
number of hidden units per layer	128
nonlinearity	ReLU
embedding size	16
λ for reachability constraint loss	1.0
learning rate	3e-4
Graph Search	
probability of using search during train	0.5
S (number of soft value iterations)	20
β (temperature)	1.1

	Point-Maze	Ant-Maze	Fetch tasks
DDPG			
Learning rate	2e-4	2e-4	1e-3
Number of workers	1	3	12
Batch size	512	1024	1024
Action L2	0.5	0.05	0.01
Gamma	0.98	0.98	0.99
Action noise	0.2	0.2	0.1
Hindsight relabelling range	80	100	50
Latent Landmarks & Auto-encoder			
Number of latent landmarks	50	50	80
Number of *warm-up trajectories*	500	500	6000
Batch size	256	256	150
Graph Search			
d_max (clipping threshold for distances)	20.0	20.0	15.0
Random landmarks added during train	150	150	20

References

Arthur, D. and Vassilvitskii, S. k-means++: The advantages of careful seeding. *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007.

Eysenbach, B., Salakhutdinov, R. R., and Levine, S. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 15246–15257, 2019.

Huang, Z., Liu, F., and Su, H. Mapping state space using landmarks for universal goal reaching. In *Advances in Neural Information Processing Systems*, pp. 1942–1952, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.