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.