Proceedings of Machine Learning Research vol 120:1–12, 2020 2nd Annual Conference on Learning for Dynamics and Control

Plan2vec: Unsupervised Representation Learning by Latent Plans

Ge Yang*† GE.IKE.YANG@GMAIL.COM
Amy Zhang*† AMYZHANG@FB.COM
Ari S. Morcos† ARIMORCOS@FB.COM
Joelle Pineau†‡ JPINEAU@CS.MCGILL.CA
Pieter Abbeel§ PABBEEL@CS.BERKELEY.EDU
Roberto Calandra† RCALANDRA@FB.COM
†*Facebook AI Research,* ‡*McGill University,* §*UC Berkeley*

Editors: A. Bayen, A. Jadbabaie, G. J. Pappas, P. Parrilo, B. Recht, C. Tomlin, M. Zeilinger

Abstract

In this paper we introduce plan2vec, an unsupervised representation learning approach that is inspired by reinforcement learning. Plan2vec constructs a weighted graph on an image dataset using near-neighbor distances, and then extrapolates this local metric to a global embedding by distilling path-integral over planned path. When applied to control, plan2vec offers a way to learn goal-conditioned value estimates that are accurate over long horizons that is both compute and sample efficient. We demonstrate the effectiveness of plan2vec on one simulated and two challenging real-world image datasets. Experimental results show that plan2vec successfully amortizes the planning cost, enabling reactive planning that is linear in memory and computation complexity rather than exhaustive over the entire state space. Additional results and videos can be found at https://geyang.github.io/plan2vec.

1. Introduction

A good representation of the state space is essential to an intelligent agent that is trying to accomplish tasks in the world. For this reason, we look at representation learning through the lens of reinforcement learning. Under the standard Markov decision process (MDP, Bellman 4) formulation, the state space S appears as the input domain for two types of functions. The first type is *local*, such as the transition probability $\mathbb{P}$ and the step-wise reward R. The second type is *non-local* and requires integration along paths, such as the state value $V(s)$ or the Q-function [12; 37; 22; 1]. For a specific type of task that can be formulated as accomplishing goals [16], the goal-conditioned value function $V(s, g)$ becomes a (negative) metric. The very focus of modern reinforcement learning is to learn V, for it parametrically encodes optimal plans. In policy search, such distance function acts are useful as a shaped reward [12; 14; 41].

In this paper, we ask the question: is there a way to learn this type of metric representation without explicitly involving interactions with the environment, using only offline exploratory data that are abundantly available? Our key insight is that one can remove the need for learning dynamics by modeling these *local* relationships between near-neighbors as the edges of a graph, then use heuristic search to generate optimal long horizon plans for path integration. Our proposed method – *plan2vec* – appears in two variants: the first uses regression towards a planned trajectory similar to dynamic distance learning [14] but with a strong search expert, whereas the second uses fitted value iteration [8; 5; 31].

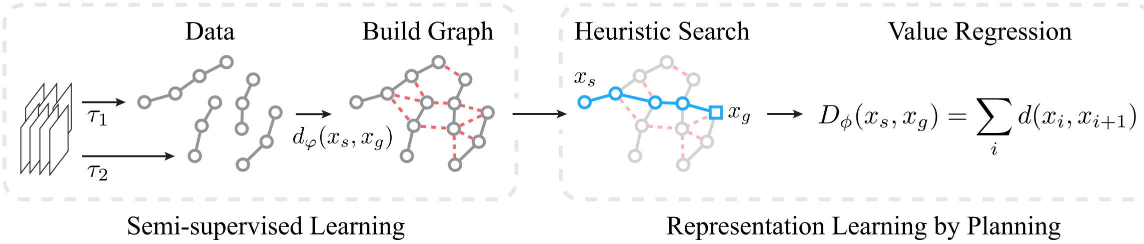

Figure 1: High-level schematics of plan2vec. The dataset contains sequences of observations. The graph building step can be considered semi-supervised learning, where the real transitions are the labeled data, and the task is to find new transitions by learning a local metric d that generalizes. The representation learning step can be considered learning an embedding ϕ of the graph. Plan2vec uses plans made on the graph to generate value targets for D, the *shortest-path-distance* metric.

To help illustrate our method, we lead the introduction of plan2vec with a set of simulated visual navigation tasks. We show the importance of using graph search as a sampling policy as opposed to a memory-less planner by looking at the planning success rate for k-steps of plan-ahead (see Fig. 6). Then we demonstrate our approach on deformable object manipulation with a rope dataset [46] that is otherwise difficult to model. Finally, we tackle a challenging real-world navigation dataset Street Learn [25] to show that plan2vec is able to learn to navigate from sequences of Street View images driving through the streets, with *no* access to the ground-truth GPS location data. Interestingly, we found that the representation plan2vec learns contains an interpretable metric map, despite that the graph it distills from is topological in nature.

2. Technical Background

The goal of reinforcement learning is to find a policy distribution $\pi(a|s)$ that maps from the state space S to the action space A for a given Markov decision process (MDP) [4], such that it maximizes the return, defined as the discounted sum of future rewards $J_\pi = \sum_t \gamma^t R(s_t, a_t)$. The optimality of a policy is provided by the Bellman equation

$$V(s) = \mathcal{T}V^*, \quad \text{where} \quad \mathcal{T}V^* \equiv R(s, a, s') + \max_a \sum_{s'} \mathbb{P}(s'|s, a)\, \gamma V^*(s'). \tag{1}$$

$\mathbb{P}(s'|s, a)$ is the transition probability. $\mathcal{T}$ is the contraction operator defined recursively on the state-value function $V(s)$. We further assume that the MDP is fully observable, so there $\exists$ a mapping $\phi(o_s) \mapsto z_s$ from the space of observations O to a latent space Z, for each state s.

When deep neural network is used as a function approximator [26], learning is typically implemented as sample-based regression towards an n-step bootstrapped value target [27]

$$\mathcal{L} = \left\| V(s_0) - \sum_{t=0}^{n} \gamma^{t-1} r - V^*(s_n) \right\|_2. \tag{2}$$

Generalized Value Function as A Metric Learning to achieve goals is an important subproblem of reinforcement learning [16]. In a goal reaching task, the agent incurs a *cost* of $-d(s, s')$ at each step. The *distance-to-goal* $D(s, g)$ refers to the *shortest path distance* $\min_\tau \sum_{x \sim \tau} d(x_i, x_{i+1})$

between s and g. This formulation offers additional structure in that D is a metric that satisfies the triangular inequality

$$\forall s' \in \mathcal{S},\ D(s, g) \leq D(s, s') + D(s', g)\,. \tag{3}$$

For this reason, the *generalized value function* (GVF, Sutton et al. 43) family of algorithms [23; 35; 19; 39] formulate learning a goal-conditioned Q value function as learning *predictive features*. For our purpose of doing unsupervised representation learning without actions, this can be simplified as learning a value $V(o, o_g) = -D_\phi(o, o_g)$ where $D_\phi(o, o_g) \equiv \|\phi(o),\ \phi(o_g)\|_p$ is the distance between the latent features vectors.

Dataset as A Graph For a dataset of images $\{x_i\}$ there $\exists$ a weighted graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ where each vertex v_i corresponds to an image x_i. $e_{ij} \in \mathcal{E}$ iff according to a local metric d, $d(x_i, x_j) < d_0$. We let the edge e_{ij} weight by $w_{ij} = d(x_i, x_j)$. If we make the additional assumption that the data are sequences of observations, then the graph is directed.

3. Unsupervised Representation Learning by Latent Planning

Plan2vec is built upon the idea that for a collection of images with a local metric d, the graph $\mathcal{G}$ weighted by d is embedded by a Riemann manifold, the metric of which is the shortest-path-distance D. By choosing the function class D_ϕ that decomposes into an embedding function $\phi(x)$ and a metric $\|\cdot\|_p$, we project D to a ℓ^p-metric space with a vector embedding.

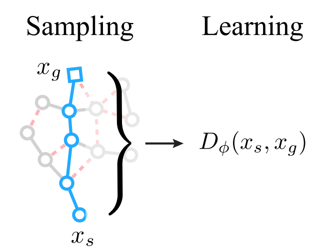

Figure 2: Plan2vec uses planning to generate value targets for the metric D_ϕ

Problem Formulation Plan2vec models the representation learning problem as learning how to play a goal-reaching-game in which one is tasked to find the shortest path from one observation to another, by hopping between near neighbors (Fig. 2). Under the context of reinforcement learning, plan2vec treats the graph as a model of the state space, and learns from Dyna-styled unroll using graph-search as an expert policy [42; 2]. Plan2vec treats the construction of the graph as a semi-supervised problem (Fig. 1). It first adds transitions from the dataset as edges with weight 1. It then uses these as labeled data to learn a local metric d, to generalize to other pairs of images as a form of loop closure. An edge e_{ij} is created between the node v_i and v_j if the distance between the corresponding images $d(x_i, x_j) \leq d_0$, a hyper parameter.

Learning Local Metric Noise-contrastive estimation cast representation learning as maximizing the contrast between two distributions: the joint distribution between related views $p(x, x^+)$, versus the product of the marginals $p(x)p(x^-)$ [28; 11; 45]

$$L_{\text{NCE}} = -\log \frac{\exp S(x, x^+)}{\exp S(x, x^+) + \sum_i^k \exp S(x, x_i^-)}\,, \tag{4}$$

where S is the similarity function to be learned. $\langle x, x^+ \rangle$ is the positive pair sampled in-context. $\langle x, x_i^- \rangle$ is the negative pair sampled independently from the marginals. Under the context of control, Eq.4 has a natural interpretation as maximizing the log-probability that a pair $\langle x, x^+ \rangle$ is *reachable* against pairs sampled at random, and is related to the *distance metric* by $d(x, x') \propto -\log p(x, x')$.

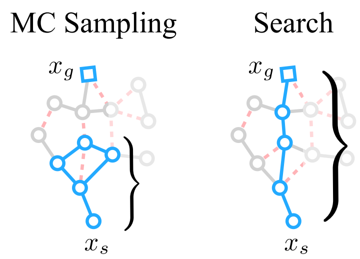

Figure 3: Memoryless sampling vs search.

Algorithm 1 Plan2vec via Amortized Search

Require: weighted directed graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$
Require: Shortest Path First search algorithm SPF
1: Initialize $D_\Phi(x, x')$, let $V = -D$.
2: **while** not converged **do**
3: sample x_{v_s}, x_{v_g} and $v_s, v_g \in \mathcal{G}$ as *start* and *goal*
4: find shortest plan $\tau^* = \mathrm{SPF}(\mathcal{G}, v_s, v_g, D_\Phi)$
5: minimize $\delta = \left| D_\Phi(x_s, x_g) - \sum_{v_i \sim \tau^*} d(x_{v_i}, x_{v_{i+1}}) \right|$

In the maze domain we directly regress the distance metric d towards one of {*identical*, *close*, or *far-apart*} with a nominal distance of $\{0,\ 1,\ 2\}$

$$\mathcal{L}_{\mathrm{d}} = |d(x, x) - 0| + |d(x_t,\ x_{t+1}) - 1| + \left|d(x,\ x^-) - 2\right|. \tag{5}$$

When d_ϕ is a Siamese network with an ℓ^2 metric, the first term can be dropped. We use smoothed L_1 loss for all terms.

Learning Representation by Latent Plans Plan2vec samples pairs of images x_s and x_g and their corresponding vertices v_s and v_g from the graph $\mathcal{G}$, then uses heuristic search to find the shortest path τ^* in-between (Algorithm 1). Non-learning search algorithms typically discard the search tree after backtrack (step 4). Plan2vec collect these to generate regression targets for learning the value estimate with or without value bootstrapping. With the latter, one can use a fixed search depth h.

$$\underbrace{\mathrm{V}(\mathrm{x_s}, \mathrm{x_g})}_{\text{no bootstrapping}} = -\sum_i d(x_{v_i}, x_{v_{i+1}}) \qquad \underbrace{\mathrm{V}(\mathrm{x_s}, \mathrm{x_g})}_{\text{bootstrapped}} = -\sum_{i=0}^{h-1} d(x_{v_i}, x_{v_{i+1}}) + V(x_{v_h}, x_{v_g}). \tag{6, 7}$$

We experimented with both fitted value-iteration (FVI) and amortized heuristic search for learning on a graph. The main short-coming with FVI is that relaxation for finding the shortest path occurs via gradient-based, iterative updates. Such scheme is unstable when applied to a graph as cycles within each rollout stall learning; whereas heuristic search explicitly avoid vertex-revisit at planning time.

4. Related Works

Plan2Vec builds upon two rich bodies of literature: unsupervised methods that learn an embedding from a local context, and value-based reinforcement learning methods that learn a policy. In the first category, time-contrastive network (TCN), skip-gram (word2vec), contrastive predictive coding (CPC) and locally linear embeddings [36; 24; 28; 32] are a family of methods that embed images, word tokens and image patches by making each sample similar to its neighbors in a small neighboring context. Similarly, graph embedding algorithms such as DeepWalk, Node2vec and diffusion maps [29; 9; 40] randomly sample short trajectories in the neighborhood of a node to provide context. The locality of such context is restrictive, because one can not expect clear supervision from samples further apart. Plan2vec solves this problem by replacing those random processes with graph-search to directly generate long-horizon distance targets between nodes that are arbitrarily far apart.

Embed to control (E2C), robust controllable embedding (RCE), L-SBMP and causal Info-GAN [48; 3; 15; 20] are a line of generative models that incorporate forward modeling in the latent space. They show that the learned representation is *plannable*, but the models are limited to modeling local relationships. Plan2vec differs by explicitly learning a *shortest-path-distance* metric to embed the weighted graph on a Riemann manifold that encodes all optimal plans as geodesics. In addition, plan2vec is purely discriminative, and focuses only on those features that are relevant towards predicting long-horizon distance relationships.

In the second category are differentiable planning algorithms on a grid world [10; 44; 21] and gradient-based planning methods that require supervision through expert demonstration [41; 50]. Plan2vec works in continuous state space, with random and off-policy exploratory data as a pre-training step. Additionally, the metric that plan2vec learns can be used as an intrinsic reward in self-supervised or task-agnostic RL [47; 7; 17; 30], to reduce the need of human designed reward.

Finally, plan2vec builds upon prior methods that plan over a graph with various assumptions [33; 34; 51; 6]. We compare against semi-parametric topological memory [33], and show that with a learned value function, plan2vec is able to make more intelligent choices at test time, under limited planning budget.

5. Experimental Evaluation

In this section, we experimentally answer the following questions: 1) What kind of representation can we learn via planning? 2) How does Dyna-style unroll on the graph affect the sample complexity? 3) Why is graph search needed? and finally, 4) Would plan2vec work in domains other than navigation, or learn features that are not visually apparent?

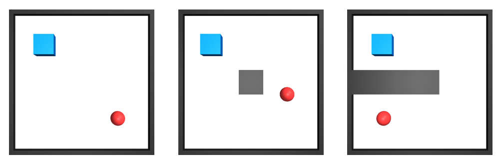

Figure 4: Visual navigation environments: *Open*, *Table*, and *C-Maze*. Agent in blue. Red sphere indicates the desired goal.

To answer these questions, we first examine plan2vec quantitatively on a simulated 2D navigation domain. Then we extend plan2vec to the challenging deformable object manipulation task, were the task is to tie a piece of rope. Finally, we show that plan2vec can learn non-visual features such as the agent's geolocation purely from first-person views without requiring ground-truth GPS data.

5.1. Simulated Navigation

The maze domain is a square, 2-dimensional arena with continuous (x, y) coordinates. A top-down camera view is fed to the robot (block in blue). We use ground-truth coordinates for evaluation only. Our experiment covers three room layouts with increasing levels of difficulty: an open room, a room with a table in the middle, and a room with a wall that separates it into two corridors that resembles a C-shaped maze (see Fig. 4).

Table 1: Planning Performance on 2D Navigation

	Success Rate (%)		
Image Input	Open Room	Table	C-Maze
plan2vec (L2)	$\mathbf{90.0 \pm 2.0}$	$\mathbf{76.4 \pm 9.2}$	$\mathbf{80.2 \pm 6.3}$
SPTM (1-step)	39.7 ± 6.1	23.7 ± 6.1	31.4 ± 6.5
VAE	73.9 ± 4.3	30.2 ± 6.5	52.7 ± 5.8
Random	3.2 ± 2.5	3.5 ± 2.5	4.7 ± 2.8

We first qualitatively verify the representation that plan2vec learns by making the latent space 2-dimensional. This allows us to directly visualize the latent vectors by plan2vec against those by a VAE ([18], see Fig. 5). The embedding VAE learns folds onto itself, whereas plan2vec learns an

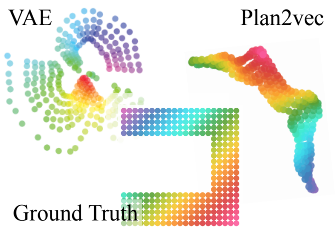

Figure 5: 2-dimensional latent embedding. Plan2vec's embedding demonstrates clear global structure beyond close neighbors.

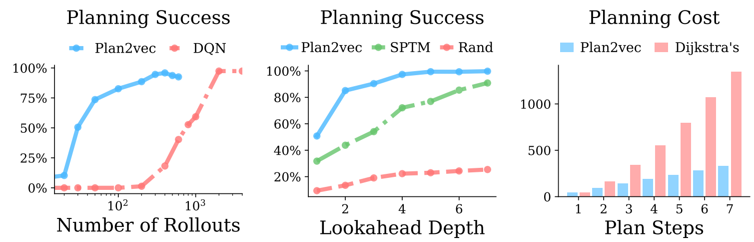

Figure 6: Left: Success rate vs the number of rollouts used for learning, Plan2vec vs DQN. Center: Success rate with k-step planahead, Plan2vec vs SPTM and a random baseline. Right: Planning Cost, Dijkstra's grows quadratically whereas plan2vec is linear. Lower is better.

embedding that respects the overall topological structure of the domain. Furthermore, observations from two opposite ends of the C-Maze are pulled apart, which reflects the longer shortest-path-distance in-between. In other words, Plan2vec embeds optimal plans as roughly straight lines in its learned latent space.

We further study how much data it takes for plan2vec to learn compared to standard off-line reinforcement methods such as fitted Q-iteration [31]. We generate a fixed dataset, then vary the amount given to both plan2vec and a standard deep Q-learning algorithm during training. We plot the planning performance of the learned value function in Fig. 6a. Both methods achieve 100% when given sufficient data, but plan2vec requires at least 1 magnitudes less. This encouraging result shows the benefit of learning from a graphical model as opposed to replays from a linear buffer, and plan2vec's ability to efficiently construct optimal plans from off-policy, exploratory experience.

Combination of search and value learning is required in harder domains that requires a strong behavior policy [38; 13; 2]. In Fig. 6b, both plan2vec and SPTM improves in performance with more lookahead search budge, but plan2vec, which distils from a search expert during training, acquires a more informative long-range value estimate and better performance. When we compare how the cost of finding the shortest path scales with the amount of planning lookahead (see Fig. 6c). We found that plan2vec is linear in plan depth, as it amortizes the planning cost from training; whereas Dijkstra's is quadratic.

5.2. Manipulation of Deformable Objects

We now apply plan2vec to learn representations of a deformable object that lacks a structured configuration space. Past methods in this space either rely on learning a generator function [20], or

Figure 7: Example of visual plan generated by plan2vec on the Rope Domain showing steps coming from two different trajectories (8 and 3). Each transition only perturbs the configuration of the rope locally. The numbers above denote the trajectory and time step the image is from, the number below represents the score by the local metric f_ϕ. Note that the transition from sequence $8 \rightarrow$ sequence 3 occurred in-between the 3rd and 4th step. All the other transitions are real physical transitions.

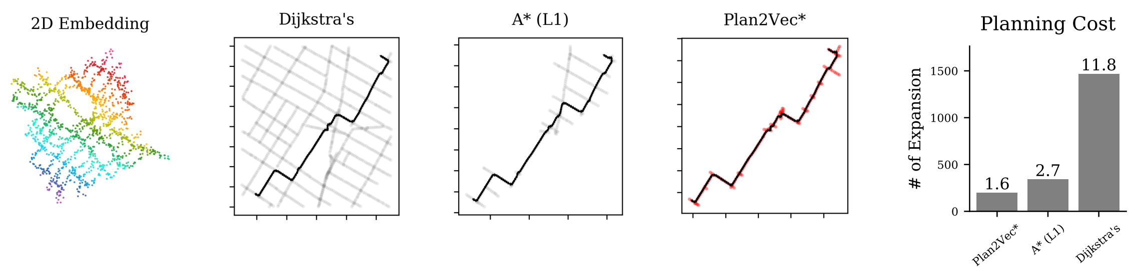

Figure 8: Planning Cost. Gray dots show the vertices that are expanded during search. We color the expanded nodes with plan2vec in red to make the few expanded nodes more visible. The plans span 200 steps, ~ 1.2 kilometers each. Number on top of bars show the average cost per planning step. With a strong heuristic (ℓ^1 distance), A* is more economical than Dijkstra's. But with a learned heuristic plan2vec approaches optimality: a single expansion per step.

model-free reinforcement learning that can only accomplish a single task [49]. In contrast, plan2vec is purely discriminative, and can generalize to a dynamic set of goals.

We apply our method to a recent rope dataset [46]. This dataset comprises of 18 sequences that include in-total 14k gray scale photos of a piece of rope. Two pegs fixiated on the table impose constraints that need to be respected during each transition. After training, plan2vec is able to find a visual plan given any pair of start and goal configurations regardless of whether they come from the same trajectory. Fig. 7 shows an example of such plans found by plan2vec. Each step only slightly perturbs the configuration of the rope, making the entire plan feasible.

It is difficult to design quantitative evaluation metrics for this domain. For evaluation, we select a start and goal image from the same trajectory, and compare the visual plans made by plan2vec against the ground-truth sequence in-between. We include these additional results in the appendix.

5.3. Beyond Visual Similarity

In previous domains, visual similarity goes a long way in revealing the distance in the configuration space. Generative models rely on such prior in order to learn, which make them potentially less suitable for learning distance information that are visually inconspicuous. Navigation in a real-world scenario offers a great example – it is impossible to tell the direction based off two photos alone. Yet a city resident knows exactly how to navigate from one to another.

Table 2: *1-step* Planning Performance on StreetLearn. Goals are sampled within 50 steps of the starting point.

	Success Rate (%)		
Street Learn	Tiny	Small	Medium
Plan2vec (Ours)	$\mathbf{92.2 \pm 2.9}$	$\mathbf{57.2 \pm 4.3}$	$\mathbf{51.4 \pm 6.9}$
SPTM (1-step)	31.5 ± 5.8	19.3 ± 5.8	20.2 ± 5.2
VAE	25.5 ± 5.6	14.4 ± 4.8	16.9 ± 5.5
Random	19.9 ± 5.4	12.0 ± 5.2	12.7 ± 4.6

We now apply plan2vec to the challenging large scale navigation dataset Street Learn [25]. We found that plan2vec's supervised learning objective can learn a high-quality value estimate on a large, $1.4k$ subset of Street Learn just under two hours, using only sequences of camera image and step-wise distance without access to the GPS locations. We inspect the learned embedding by restricting the latent space to 2-dimension, and discover a high-quality metric map (Fig. 8a).

Internalizing such a map can speed up planning and improve generalization. In Fig. 8 we compare the cost of heuristic search with and without using the distance function plan2vec learns. Dijkstra's SPF algorithm expands all nodes in the graph exhaustively, whereas A^* using the *Manhattan distance*

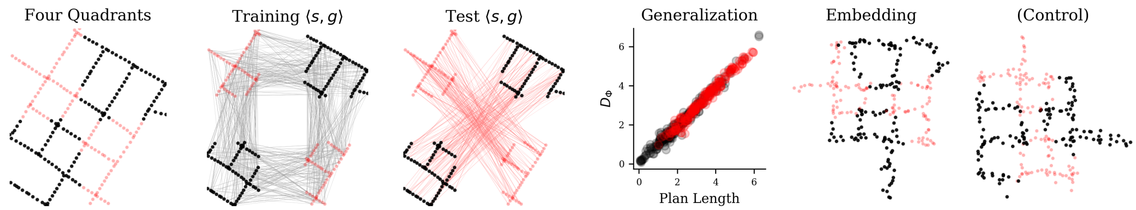

Figure 9: Plan2vec's internal metric map generalize to new tasks. (a) **Pink** and **black** color codes the four quadrants (b, c) shows the train and test tasks (d) **Red** shows the distance prediction on test set, **black** on training set. (e, f) Embedding learned using this restricted training set is similar to a control using the entire dataset. Orientation of learned embedding depends on seed. Alignment to the axis is due to use of $p = 1.2$. Randomly picked amongst 3 seeds.

as search heuristic fails to understand the diagonal streets near Broadway. Plan2vec almost optimally captures the shortest-path-distance on this domain, and out performs all other methods.

In Table 2, we artificially limit the computation and memory budget for the planner by setting both the lookahead depth k and the memory size $|\mathcal{H}|$ for the priority queue to 1. In this interesting regime, a good planning heuristic is necessary for good performance. We train an embedding $\phi(x)$ with each method, then use an ℓ^2 metric defined on this embedding as the search heuristic. The VAE baseline barely performs above random. This is expected for unsupervised methods that rely on visual inductive priors for embedding. In comparison to SPTM's 1-step local metric d, plan2vec performs 2-3$\times$ better consistently across all three datasets.

5.4. Generalization With A Metric Map

An important reason to distill plans into a neural network is generalization. In previous experiments, planning generalizes to previously unseen tasks by interpolation. Now we want to ask: how about we remove training tasks that go between large areas of the map – would plan2vec still able to generalize to bundles of task configurations it has never seen during training? In this experiment, we divide the map into four quadrants (Fig. 9a) and remove tasks that route between diagonally opposing quadrants during training. To our surprise, the learned embedding is as good as the control that trains with the entire task set. This result depends on the connectivity of the road network, but it shows that plan2vec can sometimes generalize despite of categorical removal of training tasks.

6. Conclusion

We have presented a discriminative and unsupervised approach to learn long-horizon distance relationships via planning. Our method does not generate images, is model-agnostic, and requires no access to expert action data. In comparison to model-free reinforcement learning methods that sample directly from the environment, our model-based approach makes more efficient use of otherwise disjoint trajectories. The embedding plan2vec learns encodes the shortest-path between observations as geodesics in the latent space, which reduce iterative planning to fast, parameterized lookup. We demonstrate these desirable properties on one simulated and two challenging real-world datasets, and propose plan2vec as a valuable pre-training step for reinforcement learning agents from off-line exploratory data.

References

[1] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

[2] Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems 30*, pages 5360–5370. 2017.

[3] Ershad Banijamali, Rui Shu, Mohammad Ghavamzadeh, Hung Bui, and Ali Ghodsi. Robust locally-linear controllable embedding. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

[4] Richard Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, pages 679–684, 1957.

[5] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

[6] Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. *ArXiv*, abs/1906.05253, 2019.

[7] Carlos Florensa, Jonas Degrave, Nicolas Heess, Jost Tobias Springenberg, and Martin A. Riedmiller. Self-supervised learning of image embedding for continuous control. *arXiv preprint arXiv:1901.00943*, 2019.

[8] Geoffrey J Gordon. Stable function approximation in dynamic programming. In *Machine Learning Proceedings 1995*, pages 261–268. Elsevier, 1995.

[9] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. *KDD*, 2016:855–864, August 2016. ISSN 2154-817X. doi: 10.1145/2939672.2939754.

[10] Saurabh Gupta, Varun Tolani, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. *International Journal of Computer Vision*, Oct 2019.

[11] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 297–304, 2010.

[12] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[13] Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Tobias Pfaff, Theophane Weber, Lars Buesing, and Peter W Battaglia. Combining q-learning and search with amortized value estimates. *arXiv preprint arXiv:1912.02807*, December 2019.

[14] Kristian Hartikainen, Xinyang Geng, Tuomas Haarnoja, and Sergey Levine. Dynamical distance learning for unsupervised and semi-supervised skill discovery. *arXiv preprint arXiv:1907.08225*, 2019.

[15] Brian Ichter and Marco Pavone. Robot motion planning in learned latent spaces. *arXiv preprint arXiv:1807.10366*, 2018.

[16] Leslie P Kaelbling. Learning to achieve goals. *IJCAI*, 1993. ISSN 1045-0823.

[17] Gregory Kahn, Adam Villaflor, Bosen Ding, Pieter Abbeel, and Sergey Levine. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. *arXiv preprint arXiv:1709.10489*, 2017.

[18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[19] Tejas D Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J Gershman. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*, 2016.

[20] Thanard Kurutach, Aviv Tamar, Ge Yang, Stuart J Russell, and Pieter Abbeel. Learning plannable representations with causal infogan. In *Advances in Neural Information Processing Systems*, pages 8747–8758, 2018.

[21] Lisa Lee, Emilio Parisotto, Devendra Singh Chaplot, Eric Xing, and Ruslan Salakhutdinov. Gated path planning networks. *arXiv preprint arXiv:1806.06408*, 2018.

[22] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[23] Michael L Littman, Richard S Sutton, and Satinder P Singh. Predictive representations of state. *Advances in Neural Information Processing Systems*, 2001.

[24] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[25] Piotr Mirowski, Matthew Koichi Grimes, Mateusz Malinowski, Karl Moritz Hermann, Keith Anderson, Denis Teplyashin, Karen Simonyan, Koray Kavukcuoglu, Andrew Zisserman, and Raia Hadsell. Learning to navigate in cities without a map. In *Advances in Neural Information Processing Systems*, 2018.

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[27] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 1928–1937, June 2016.

[28] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[29] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[30] Vitchyr H. Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

[31] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

[32] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[33] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. *arXiv preprint arXiv:1803.00653*, 2018.

[34] Nikolay Savinov, Anton Raichuk, Raphaël Marinier, Damien Vincent, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic Curiosity through Reachability. *arXiv preprint arXiv:1810.02274*, 2018.

[35] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning (ICML)*, pages 1312–1320, 2015.

[36] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141, 2018.

[37] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning (ICML)*, 2014.

[38] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, October 2017. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature24270.

[39] Satinder P Singh, Michael L Littman, Nicholas K Jong, David Pardoe, and Peter Stone. Learning predictive state representations. *International Conference on Machine Learning (ICML)*, 2003.

[40] Richard Socher and Matthias Hein. Manifold learning and dimensionality reduction with diffusion maps. In *Seminar report, Saarland University*, 2008.

[41] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In *International Conference on Machine Learning (ICML)*, volume 80, pages 4732–4741, 10–15 Jul 2018.

[42] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, July 1991. ISSN 0163-5719. doi: 10.1145/122344.122377.

[43] Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 761–768. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[44] Aviv Tamar, YI WU, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 2154–2162. 2016.

[45] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

[46] Angelina Wang, Thanard Kurutach, Aviv Tamar, and Pieter Abbeel. Learning robotic manipulation through visual planning and acting. In *Deep RL Workshop at NeurIPS*, 2018.

[47] David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. *arXiv preprint arXiv:1811.11359*, November 2018.

[48] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pages 2746–2754, 2015.

[49] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to manipulate deformable objects without demonstrations. *arXiv preprint arXiv:1910.13439*, October 2019.

[50] Tianhe Yu, Gleb Shevchuk, Dorsa Sadigh, and Chelsea Finn. Unsupervised visuomotor control through distributional planning networks. *arXiv preprint arXiv:1902.05542*, 2019.

[51] Amy Zhang, Sainbayar Sukhbaatar, Adam Lerer, Arthur Szlam, and Rob Fergus. Composable planning with attributes. In *International Conference on Machine Learning (ICML)*, pages 5837–5846, 2018.