Learning Navigation Costs from Demonstration via Differentiable Planning

Tianyu Wang
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, U.S.A.
tiw161@eng.ucsd.edu

Nikolay Atanasov
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, U.S.A.
natanasov@eng.ucsd.edu

***Abstract*—This paper focuses on learning cost functions that capture desirable behavior in tasks demonstrated by an expert. In a task such as autonomous navigation in unknown environments, it is possible to obtain sequences of observations (e.g., images), states (e.g., poses), and control inputs but not direct queries of the underlying task-specific cost function. Hence, it is necessary to design a planning algorithm that, depending on the current cost representation, computes a control policy and propagates its error with respect to the demonstrations back to the cost representation. Our contribution is a probabilistic environment representation for local observation updates and cost function design, and a differentiable planning algorithm that performs state expansions only on a subset of promising states. Our complete model can be trained end-to-end and improves upon Value Iteration Networks and the Dyna-Q algorithm.**

***Index Terms*—Inverse Reinforcement Learning, Differentiable Planning, Cost Learning**

I. Introduction

Autonomous robot systems increasingly require operation in unstructured, partially known, and dynamically changing environments. One core challenge for safe and robust navigation is that the *true* cost function of a navigation task, requiring safe, dynamically feasible, and efficient behavior, is generally not known while expert demonstrations can be utilized to uncover the underlying cost function [1], [2]. In addition, humans and animals can navigate successfully with partial knowledge of the environment and adapt when facing new obstacle configuration based on prior experience. Motivated by this observation, we focus on learning a cost function from demonstration that is not universally accurate over the state and control space but rather captures task-relevant information and leads to desirable behavior.

Our main contribution is an end-to-end differentiable model that combines a cost function representation and an efficient planning algorithm (see Fig. 1). The novelty of our approach is that the proposed model is fully differentiable, which allows using gradient-based optimization to improve the parameterized cost function. Our experiments show that the end-to-end differentiable model learns task-specific cost functions and improves upon Value Iteration Networks (VIN) [3] and the Dyna-Q algorithm [4] by handling partial and noisy observations. In summary, we offer the following contributions:

We gratefully acknowledge support from NSF CRII RI IIS-1755568.

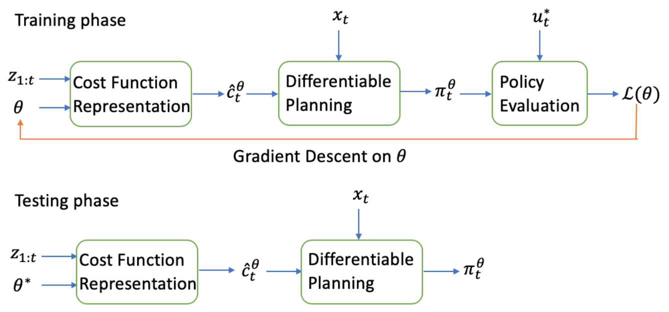

Fig. 1: Architecture for learning cost function representations from demonstrations via differentiable planning. During training, the goal is to learn a cost function parameterization θ based on demonstrations, consisting of states $x_{1:t}$, controls $u^*_{1:t}$, and partial observations $z_{1:t}$, so that a control policy generated based on the learned cost incurs minimum loss $\mathcal{L}(\theta)$. Both the cost function representation $\hat{c}_t^\theta$ and the control policy π_t^θ need to be differentiable with respect to θ for error backpropagation. During testing in a new environment, online observations $z_{1:t}$ and the trained parameters θ^* provide the cost function necessary to generate a control policy.

- A cost function representation that incorporates a log-odds occupancy representation of the environment, updatable using a parameterized observation model.
- An efficient planning algorithm, which performs local convolutional operations encoding Bellman backups *only* on a subset of promising states. We guarantee that the output policy is differentiable with respect to the input cost function.
- An end-to-end differentiable model that learns task-specific cost functions from expert demonstrations by backpropagating policy performance loss through the planning algorithm and the cost representation.

II. Problem Formulation

Consider a robot navigating in an unknown environment with the task of reaching a goal state $x_g \in \mathcal{X}$. Let $x_t \in \mathcal{X}$ be the discrete time robot state. For a given control input $u_t \in \mathcal{U}$, the robot state evolves according to known deterministic dynamics: $x_{t+1} = f(x_t, u_t)$. Let m^* be a function

Algorithm 1 Differentiable A* Planning

```
1:  **Input:** Current state $x_t$, goal state $x_g$, cost function $\hat{c}_t^\theta$, heuristic function
    $h$, transition function $f$
2:  $g(x_g) = 0,\ g(x) = \infty\ \forall x \in \mathcal{X}$
3:  OPEN $\leftarrow \{x_g\}$, CLOSED $\leftarrow \{\}$
4:  **while** $x_t \notin$ CLOSED **do**
5:      Remove $x$ with smallest $g(x) + h(x)$ value from OPEN and insert
        $x$ in CLOSED
6:      **for** $u \in \mathcal{U}$ **do**
7:          $g(y) \leftarrow \textsc{Conv}(g(s), \delta(f(s,u) - x))$
8:          $c(y,u) \leftarrow \textsc{Conv}(\hat{c}_t^\theta(s,u), \delta(f(s,u) - x))$
9:          $g(y) \leftarrow \min\{g(y), c(y,u) + g(x)\}$
10:     **end for**
11: **end while**
12: $Q(x_t, u_t) \leftarrow \hat{c}_t^\theta(x_t, u_t) + g(f(x_t, u_t))\ \forall u_t \in \mathcal{U}$
13: **return** policy $\pi_t^\theta(u_t|x_t) = \frac{\exp(-Q(x_t,u_t))}{\sum_{u' \in \mathcal{U}} \exp(-Q(x_t,u'))}$
```

specifying the *true* occupancy of the environment and let c^* be a cost function specifying desirable robot behavior in a given environment. We assume that the robot does not have access to either m^* or c^* but is able to make noisy observations $z_t \in \mathcal{Z}$ of the environment in its vicinity. Given a training set $\mathcal{D} := \left\{(x_{t,n}, u_{t,n}^*, z_{t,n})\right\}_{t=1,n=1}^{T_n,N}$ of N expert trajectories with length T_n, the robot's goal is to learn a cost function estimate $\hat{c}_t^\theta : \mathcal{X} \times \mathcal{U} \times \mathcal{Z}^t \to \mathbb{R}_{\geq 0}$ parameterized by θ and execute a new navigation task by computing a stochastic control policy $\pi_t^\theta(\cdot|x_t)$ based on $\hat{c}_t^\theta$, mapping from a state x_t to a control distribution.

The problem setup is illustrated in Fig. 1. The challenge is to make the optimization of the policy π_t^θ differentiable with respect to the cost function parameterization θ so that the loss can be backpropagated through the planning algorithm.

III. Technical Approach

A. Occupancy-based Cost Function Representation

We propose an cost function representation based on occupancy grid maps. The state space $\mathcal{X}$ is discretized into N cells and the robot observes binary measurements of the K cells around it, i.e., $z_t \in \{-1, 1\}^K$. We model the field of view of the robot using a binary matrix $H(x_t) \in \{0,1\}^{K \times N}$ so that $y_t := H(x_t)m^*$ are the true occupancy states of the observable cells around the robot state x_t. We can explicitly design a parameterized observation model $p(z_t \mid x_t, m; \theta) = \prod_{i=1}^K p(z_t[i] \mid y_t[i]; \theta_{1,t}[i])$ using a neural network or simply a sigmoid function. The estimated probability mass function of the map occupancy $p_t^\theta(m) := p(m \mid z_{1:t})$ can be updated as new observations arrive and and a cost function $\hat{c}_t^\theta$ can be induced based on the occupancy probability of each cell.

B. Differentiable A* Planning

Next, we develop a differentiable A* algorithm (see Alg. 1), which performs efficient convolutions only over a subset of promising states instead of over the entire state space. Specifically, we can initialize the A* algorithm from x_g, search for an optimal path backwards to the current state x_t, and, at termination, the g-values will provide an upper bound to the optimal cost-to-go for each state [5] (the upper bound is tight for those in the CLOSED list). We can make the A* algorithm differentiable by using convolutional and minpooling operations when updating the g-values. The differentiability is guaranteed in line 7 and 8, obtaining the values $g(y)$ and $c(y,u)$ at a predecessor state y as convolution with a delta kernel $\delta(f(s,u) - x)$, representing a deterministic transition from y to $f(y,u)$, and in line 7, updating $g(y)$ by min-pooling between the quantities $g(y)$ and $\hat{c}_t^\theta(y,u) + g(x)$.

Models	Succ. rate	Traj. diff.
VIN	95.12 %	0.264
Ours-A*	100 %	0.183

Fig. 2: Test set performance of VIN and our models.

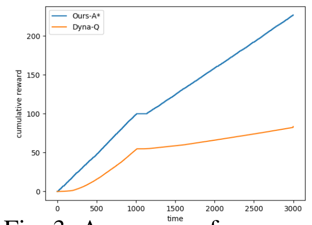

Fig. 3: Average performance of **Dyna-Q** and **Ours-A***

IV. Evaluation

A. 2-D Grid Navigation

The first experiment evaluates whether the models can make successful navigation in a 2D gridworld environment with randomly generated obstacles. We consider the fully observable environment where the observation is the map and the expert controls are calculated by Dijkstra's algorithm. We compare **Ours-A*** with **VIN** [3]. Fig. 2 shows the navigation performance in terms of successfully reaching the goal and discrepancy between robot planning trajectory and the optimal one.

B. Online Learning in Changing Environments

In this experiment, we compare **Ours-A*** with the **Dyna-Q** [4] algorithm in a changing environment where the optimal path is switched from one to another in the middle of an episodic task. The robot can only receive partial observation at the next state that it plans to arrive at. Fig. 3 shows the performance of the **Dyna-Q** and **Ours-A*** averaged over 10 trials. The higher the slope of the curve, the better the model is in finding a shortest path to goal. **Ours-A*** plateaus when the environment changes but quickly adapts to the new configuration and consistently finds the new path afterwards.

References

[1] B. D. Ziebart, A. Maas, J. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, 2008, pp. 1433–1438.

[2] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 663–670.

[3] A. Tamar, Y. WU, G. Thomas, S. Levine, and P. Abbeel, "Value iteration networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 2154–2162.

[4] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[5] D. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.