Methods in Large Scale Inverse Optimal Control

by

Mathew Monfort
B.A. Mathematics (Franklin and Marshall College) 2009
M.S. Computer Science (Florida State University) 2011

Thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2016

Chicago, Illinois

Defense Committee:
Brian Ziebart, Chair and Advisor
Tanya Berger-Wolf
Piotr Gmytrasiewicz
Lev Reyzin, Mathematical Computer Science
Peter Carr, Disney Research Pittsburgh

For Yella

ACKNOWLEDGMENTS

I attribute the quality of the work in this thesis to my Advisor Brian Ziebart and the Purposeful Prediction Lab for allowing me to work on a large variety of problems while a student at UIC. I also want to thank my collaborators without whom much of the work in this thesis would not have come to fruition: Anqi Liu, Xiangli Chen, Brenden Lake, Arunkumar Byravan, Patrick Lucey, Matthew Johnson, Katja Hofmann, Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Lawrence D Jackel, Urs Muller, Xin Zhang, Karol Zieba, Timothy Luciani, Liz Marai, Sima Behpour, Christopher Schultz, Kaiser Asif and Xiuwen Liu.

CONTRIBUTION OF AUTHORS

Chapter 1 intoroduces the general motivation and problem this document attempts to solve and was written solely by me. Chapter 2 introduces related work to provide the broader scientific context of this document and was written solely by me. Chapter 3 is a previously presented in a workshop (51) but not formally published in any proceedings or journal. I developed the conceptual framework. My advisor, Brian Ziebart, guided this project and provided very helpful feedback throughout the process. I was the primary author. Chapter 4 is a published manuscript (50) for which I developed the conceptual framework and wrote the manuscript in close collaboration with my advisor Brian Ziebart. I was the primary author. Chapter 5 provides an overview and analysis of the results generated using the methods described in chapters 3 and 4. A large portion of this chapter is from a published manuscript (50) where I developed the conceptual framework, designed and carried out the experiment, and performed the analysis. The experiment was carried out on handwriting data collected by Brenden M. Lake and Josh Tenenbaum and professional soccer data provided by Patrick Lucey. I was the primary author of the paper. Chapter 6 is a published manuscript (52) for which I developed the conceptual framework, designed and carried out the experiment, performed the analysis, and wrote the manuscript in close collaboration with my advisor Brian Ziebart and Anqi Liu. I was the primary author of the paper. Chapter 7 is partially work previously presented in a workshop (13) but not formally published in any proceedings or journal and work from a published manuscript (12) of which I was a contributing author. I developed the conceptual framework, designed and car-

CONTRIBUTION OF AUTHORS (Continued)

ried out the experiment, and performed the analysis for all sections pertaining to the waypoint guided linear quadratic regulation method that the chapter focusses on. The remainder of the papers were completed with Arunkumar Byravan as the primary author with assistance from myself, Brian Ziebart, Byron Boots, and Dieter Fox. Chapter 8 concludes the document with a discussion of the presented methods and future directions of this work and was written solely by me.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

LIST OF TABLES

LIST OF FIGURES

SUMMARY

As our technology continues to evolve, so does the complexity of the problems that we expect our systems to solve. The challenge is that these problems come at increasing scales that require innovative solutions in order to be tackled efficiently. The key idea behind Inverse Optimal Control (IOC) is that we can learn to emulate how a human completes these complex tasks by modeling the observed decision process. This thesis presents algorithms that extend the state-of-the art in IOC in order to efficiently learn complex models of human behavior.

We explore the use of an admissible heuristic in estimating path distributions through weighted graphs. This includes a modified version of the softened policy iteration method used in Maximum Entropy Inverse Optimal Control and present the *SoftStar* algorithm which merges ideas from Maximum Entropy IOC and A* Search for an efficient probabilistic search method that estimates path distributions through weighted graphs with approximation guarantees.

We then explore IOC methods for prediction and planning in problems with linear dynamics that require real-time solutions. This includes an inverse linear quadratic regulation (LQR) method for efficiently predicting intent in 3-dimensional space and a discrete-continuous hybrid version of inverse LQR that uses discrete waypoints to guide the continuous LQR distribution.

The presented techniques are evaluated on a number of different problem settings including planning trajectories of handwritten characters, modeling the ball-handler decision process

SUMMARY (Continued)

in professional soccer, predicting intent in completing household tasks, and planning robotic motion trajectories through a cluttered workspace.

CHAPTER 1

INTRODUCTION

Partially published in the Proceedings of the Neural Information Processing Systems Conference (https://papers.nips.cc/paper/5889-softstar-heuristic-guided-probabilistic-inference) (50).

The mechanical and cognitive properties of the human body and mind consistently bias the way that we solve the problems we encounter. Human behavior is much more structured than physical limitations require; variability in tasks ranging from manipulating an object (14) to locomoting (70) is relatively small. In order to model human behavior we must first develop systems that can learn this structure within the associated decision processes.

Inverse optimal control (IOC) (33), also known as inverse reinforcement learning (54; 1) and inverse planning (5), has become a powerful technique for learning to control or make decisions based on expert demonstrations (1; 63). Rather than directly imitating the demonstrated control policy by learning a mapping from state to control (i.e., behavioral cloning) (61), IOC estimates an underlying feature-based utility function that motivates the observed behavior (rationalizes an expert's demonstrated control sequences) (54). This estimated utility function produces solution policies that typically generalize across different decision processes far better than directly estimated policies. These estimated utilities can then be used in an (optimal) controller to solve new decision problems, producing behavior that is similar to demonstrations.

Predictive extensions to IOC (53; 81; 4; 52; 62; 10) recognize the inconsistencies, and inherent sub-optimality, of repeated behavior by incorporating uncertainty. For instance, Max-

imum Entropy IOC (81) methods estimate a stochastic policy that is most uncertain while still guaranteeing the same expected cost as demonstrated behavior on an unknown cost function (1). This allows for the optimal path to have the highest probability with non-optimal paths becoming exponentially less likely under a stochastic policy that is dependent on the expected cumulative cost of the succeeding trajectory. These methods provide probabilistic forecasts of future decisions in which stochasticity is due to this uncertainty rather than the stochasticity of the decision process's dynamics. These models' distributions over plans and policies can typically be defined as softened versions of optimal sequential decision criteria.

A key challenge for predictive IOC is that many decision sequences of interest are embedded within exceedingly large decision processes. Additionally, standard IOC requires solving the 'forward' task of finding the optimal policy for each update of the cost function. This is needed to calculate the feature expectations necessary to update the cost function parameters. This becomes expensive in large-scale problems with complex decision spaces. Symmetries in the decision process can be exploited to improve efficiency (75), but decision processes are not guaranteed to be (close to) symmetric. Maximum Margin Planning (63) address the problem of learning complex behavior by framing the problem as a maximum margin classifier allowing for an optimal control solution to be used as a sub-routine rather than integrating over the set of possible sequences in Maximum Entropy IOC (81), however it makes the assumption that the demonstrated behavior is optimal and can be expressed via a single function. Local optimality can also be exploited for efficiently learning complex continuous sequences of behavior (46) by locally approximating the probability distribution, but the assumption of a

deterministic fixed horizon problem and the neglect of global optimality in the demonstrated behavior limits its practicality in many complex tasks.

This thesis develops methods that address the problem of increased complexity and intractable solutions for inverse optimal control in large decision processes in both discrete and continuous settings.

We consider discrete and continuous decision process representations in our proposed methods. For the discrete case, we incorporate ideas from A* search into the Maximum Entropy IOC (82) framework in order to efficiently approximate the probable behavior patterns resulting from the estimated utility function. This allows for us to accurately model control policies in large scale decision processes with reasonable complexity and approximation guarantees in discrete settings.

We then introduce a continuous IOC method (Inverse Linear Quadratic Regulation) for developing real-time predictions in complex settings. This is beneficial for any task that requires interaction with the modeled system (e.g., human-robot interaction). We extend this method further to incorporate discrete paths, which can be generated from a discrete IOC model, that bind the predicted/planned continuous trajectory distributions as waypoints guiding the LQR system.

The presented methods each address a specific problem related to modeling large scale decision processes including discrete sampling, discrete searching, continuous prediction and hybrid discrete-continuous planning. The main goal of this work is to develop techniques that

will allow for more complex behavior models to be formed as the scale of the problems being discovered increase in complexity.

1.1 Overview

This thesis is separated into three parts: Introduction and Background, Approximating Path Distributions in Weighted Graphs, and Inverse Linear Quadratic Regulation. This separation helps to organize the work into two major areas of contribution: discrete and continuous IOC.

The first part (Introduction and Background) introduces the general problems associated with large scale inverse optimal control and includes a background chapter describing any pre-requisite knowledge that may be needed in order to gain a full understanding of the rest of the thesis.

The second part (Approximating Path Distributions in Weighted Graphs) explores the use of an admissible heuristic to aid in estimating near optimal path distributions through weighted graphs. This includes a modified version of the softened value iteration method used in Maximum Entropy Inverse Optimal Control (IOC) (82) and introduces the *SoftStar* algorithm which merges ideas from Maximum Entropy IOC and A* Search (29) in order to form an efficient probabilistic search algorithm that estimates path distributions through weighted graphs with approximation guarantees. We end this part with a chapter highlighting the experimental results of the proposed algorithms in two complex discrete decision problems.

The third part (Inverse Linear Quadratic Regulation) focuses on continuous IOC methods for prediction and planning in problems that require real-time solutions under assumed linear dynamics. We begin with a chapter detailing an inverse linear quadratic regulation (LQR)

method for efficiently predicting intent and forecasting future trajectories in three dimensional space. We then extend that idea in the following chapter to incorporate discrete trajectories in the form of waypoints that guide the continuous LQR distribution. Both of the chapters in this part include experimental results on complex problem settings requiring the unique benefits of each algorithm respectively.

We then finish with a final chapter detailing closing remarks, directions for future research, and general thoughts regarding the presented material.

CHAPTER 2

BACKGROUND AND RELATED WORK

Partially published in the Proceedings of the Neural Information Processing Systems Conference (https://papers.nips.cc/paper/5889-softstar-heuristic-guided-probabilistic-inference) (50).

2.1 Markov Decision Processes

A Markov decision process is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathrm{R})$ of states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, a probabilistic state transition mapping $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, and a reward function $\mathrm{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ mapping state transitions to reward values. A policy $\pi : \mathcal{S} \to \mathcal{A}$ maps states to actions and represents a decision making process[1]. The optimal policy maximizes the expected cumulative reward,

$$\pi^*(s_i) = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_\pi\Big[\sum_{t=i}^{T} \mathrm{R}(s_t, a_t)\Big]. \tag{2.1}$$

[1] More generally, a policy estimates the probability distribution over possible actions, $\Pi : \mathcal{S} \to \Delta\mathcal{A}$, which is then sampled to select a particular action

In practice, the anticipated reward of taking action a_t when in state s_t is represented by a state-action value function $Q(s_t, a_t)$. Maximizing over the set of possible actions results in the state value function and optimal policy,

$$V(s_t) = \max_{a_t} Q(s_t, a_t), \tag{2.2}$$

$$\pi(s_t) = \operatorname*{argmax}_{a_t} Q(s_t, a_t). \tag{2.3}$$

Equivalently, the optimal policy satisfies the Bellman equation (9),

$$Q(s_t, a_t) = \mathbb{E}[V(s_{t+1})|s_t, a_t] + R(s_t, a_t), \tag{2.4}$$

which can be obtained with value iteration—a dynamic programming algorithm that iteratively computes $V(s_t)$ and $Q(s_t, a_t)$ to obtain the cumulative expected rewards of the optimal policy when starting from state s_1.[1]

2.2 State-space graphs

The space of plans and their costs can be succinctly represented using a state-space graph, $\mathcal{G} = (\mathcal{S}, \mathcal{E}, \mathrm{cost})$. Vertices, $s \in \mathcal{S}$, represent states of the planning task and directed edges, $e_{ab} \in \mathcal{E}$, represent available transitions between states corresponding to vertices s_a and s_b. The neighbor set of vertex s, $\mathcal{N}(s)$, is the set of vertices to which s has a directed

[1] Dijkstra's algorithm (20) is a similar dynamic program for finding optimal paths for decision processes with deterministic dynamics.

edge. A cost function, $cost(s, s')$, represents the relative desirability of transitioning between states s and s', and can be incorporated into the state-space graph on each edge.

The optimal plan from state s_1 to goal state s_T is a variable-length sequence of states, $(s_1, s_2, \ldots, s_T)$ forming a path through the state-space graph that minimizes a cumulative penalty. Letting $h(s)$ represent the cost of the optimal path from state s to the goal state s_T (i.e., the cost-to-go or value of s) and defining $h(s_T) \triangleq 0$, the optimal path corresponds to a fixed-point solution of the next state selection criterion (9):

$$h(s) = \min_{s' \in \mathcal{N}(s)} h(s') + cost(s, s'), \tag{2.5}$$

$$s_{t+1} = \operatorname*{argmin}_{s' \in \mathcal{N}(s_t)} h(s') + cost(s_t, s'). \tag{2.6}$$

The optimal path distance from the start state, $d(s)$, can be similarly defined (with $d(s_1) \triangleq 0$) as

$$d(s) = \min_{s' : s \in N(s')} d(s') + cost(s', s). \tag{2.7}$$

Dynamic programming algorithms, such as Dijkstra's algorithm (20), search the space of paths through the state-space graph in order of increasing $d(s)$ to find the optimal path. Doing so implicitly considers all paths up to the length of the optimal path to the goal state.

2.3 Heuristic-guided search

Additional knowledge can significantly reduce the portion of the state space needed to be explored to obtain an optimal plan. For example, A* search (29) explores partial state sequences by expanding states that minimize an estimate,

$$\mathrm{f}(\mathrm{s}) = \mathrm{d}(\mathrm{s}) + \widehat{\mathrm{h}}(\mathrm{s}), \tag{2.8}$$

combining the minimal cost to reach state s, $\mathrm{d}(\mathrm{s})$, with a heuristic estimate of the remaining cost-to-go $\widehat{\mathrm{h}}(\mathrm{s})$.

When the heuristic estimate is admissible (i.e., a lower bound, $\widehat{\mathrm{h}}(\mathrm{s}) \leq \mathrm{h}(\mathrm{s})$ for all $\mathrm{s} \in \mathcal{S}$), the algorithm can terminate with the guaranteed optimal solution once the best "unexpanded" state's estimate, $\mathrm{f}(\mathrm{s})$, is worse than the best discovered path to the goal state.

Algorithm 1 A* Search with admissible heuristic

Input: State-space graph $\mathcal{G}$, start state s_1, goal state s_T, heuristic function $\hat{h}$

Output: Optimal path cost

Set $d(s) = \infty$ for all $s \in \mathcal{S}$ Insert $(s_1, \hat{h}(s_1))$ into priority queue P

while $d(s_T) \geq$ *min estimate in P* **do**

  $(s, f(s)) \leftarrow$ state/estimate of min element popped from P

  if $(f(s) - \hat{h}(s) < d(s))$ **then**

    Set $d(s) = f(s) - \hat{h}(s)$

    for $s' \in N(s)$ **do**

      (Re-)Insert $(s', d(s) + cost(s, s') + \hat{h}(s'))$ in P

    end

  end

end

return $d(s_T)$

A* search using an admissible heuristic is described in Algorithm 1. Note that if s' already exists in the priority queue, its estimate can be updated to the minimum of the new insertion estimate and previous estimate rather than including repeated states with different estimates in the priority queue. Any additions that are worse than previously expanded states can likewise be discarded. Further, when the heuristic function is monotonic, subsequent exploration (i.e., popping from the priority queue) of the same state in A* (Algorithm 1) will always have larger

distances. Thus, the A* algorithm can terminate when the goal state is explored, and, more generally, only states that have not already been explored need to be added to the queue (18).

Extensions to A* for parametric planning leverage the concavity of weighted multi-criterion planning objective functions to efficiently provide optimal plans for different trade-offs in plan criteria (80).

2.4 Inverse Reinforcement Learning

Reinforcement learning (69) is the process by which an agent finds the optimal policy given a reward function in an unknown environment. If we can consider Equation 2.4 with a given reward function and unknown stochastic dynamics, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S}' \rightarrow \mathcal{P}(\mathcal{S}'|\mathcal{S}, \mathcal{A})$,

$$\begin{aligned} Q(s_t, a_t) &= \sum_{s_{t+1} \in \mathcal{S}} p(s_{t+1}|s_t, a_t) V(s_{t+1}) + R(s_t, a_t) \\ &= \mathbb{E}[V(s_{t+1})|s_t, a_t] + R(s_t, a_t), \end{aligned} \tag{2.9}$$

the goal of reinforcement learning is to find the optimal policy that maximizes the given reward function.

Inverse reinforcement learning (56) optimizes the reward function with the goal of matching the policy found by reinforcement learning to the observed policy in a set of demonstrated behavior. An initial reward function is used to generate a policy which is then evaluated and compared to the demonstrated policy. Using this information, the reward function is then updated in order to reduce the error between the value of the demonstrated policy and that of the policy dictated by the current reward function, $\left[V^*(s) - V^{\pi}(s)\right]$.

2.5 Maximum Entropy Inverse Optimal Control

Inverse Optimal Control (IOC) (33; 11; 55) estimates the reward/cost function that makes demonstrated sequential behavior optimal in a known environment (given transition dynamics). Abbeel and Ng (2) solved this problem by constructing a distribution over optimal policies that matches *feature counts* with demonstrated behavior:

$$\mathbb{E}_{\hat{p}}\left[\sum_{t=1}^{T}\mathbf{f}(s_t,a_t)\right]=\mathbb{E}_{\tilde{p}}\left[\sum_{t=1}^{T}\mathbf{f}(s_t,a_t)\right]. \tag{2.10}$$

When we model the unknown cost function as a linear function of these features, $\mathbf{f}$, and a set of learned parameters, θ,

$$\mathrm{cost}(s_t,a_t)\triangleq\theta^{T}\mathbf{f}(s_t,a_t), \tag{2.11}$$

matching feature counts guarantees that the estimated control policy (mixture) will realize the same expected cost as the demonstrated behavior.

Maximum entropy IOC algorithms (81; 78) achieve this by obtaining a stochastic action policy that is most uncertain while still guaranteeing the same expected cost as demonstrated behavior on an unknown cost function (1). This is done by obtaining the stochastic policy that is least biased while still satisfying Equation 2.10. Maximum entropy IOC allows for the optimal path to have the highest probability with non-optimal paths becoming exponentially less likely

under a stochastic policy that is dependent on the expected cumulative cost of the succeeding trajectory,

$$\hat{P}(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}) = \frac{1}{Z} e^{-\sum_t^T \text{cost}(\mathbf{s}_t, \mathbf{a}_t)}, \tag{2.12}$$

where Z is the partition function that requires knowledge of the complete policy under the current cost function.

Calculating the marginal state probabilities of this distribution is important for making predictions and estimating model parameters. The forward-backward algorithm (6) can be employed, but for large state-space graphs it may not be practical.

To ensure the maximum entropy conditions are satisfied, we replace the $\max(\cdot)$ operation in (Equation 2.2) with a continuous relaxation (the *softmax*) using the log-sum-exponential expression (78),

$$V(\mathbf{s}_t) = \underset{\mathbf{a} \in \mathbf{A}_{\mathbf{s}_t}}{\text{softmax}}\, Q(\mathbf{s}_t, \mathbf{a}_t) \triangleq \log \sum_{\mathbf{a} \in \mathbf{A}_{\mathbf{s}_t}} e^{Q(\mathbf{s}_t, \mathbf{a}_t)}. \tag{2.13}$$

With a probabilistic policy formed using a Gibbs distribution (rather than a deterministic policy mapping states to actions),

$$\hat{\pi}(a_t | s_t) = e^{Q(s_t, a_t) - V(s_t)}. \tag{2.14}$$

The optimization problem is to learn a cost function (Equation 2.11) that explains the observed behavior preferences of a given set of demonstrations. We can solve this by first forming the objective function,

$$L(\theta) = \mathbb{E}_{\hat{\pi}}\Big[V(\mathbf{s}_1)\Big] - \mathbb{E}_{\tilde{\pi}}\Big[\sum_{t=1}^{\tilde{T}} \theta^T f(\mathbf{s}_t, \mathbf{a}_t)\Big], \tag{2.15}$$

and then updating the weight vector, θ, with a gradient derived from Equation 2.15 which computes the difference in the feature expectations found using the computed policy, $\hat{\pi}$, and those of the demonstrated policy, $\tilde{\pi}$ (81),

$$\nabla L(\theta) = \mathbb{E}_{\hat{\pi}}\Big[\sum_{t=1}^{\hat{T}} f(\mathbf{s}_t, \mathbf{a}_t)\Big] - \mathbb{E}_{\tilde{\pi}}\Big[\sum_{t=1}^{\tilde{T}} f(\mathbf{s}_t, \mathbf{a}_t)\Big]. \tag{2.16}$$

2.6 Approximate Maximum Entropy Inverse Optimal Control

An approximation of Maximum Entropy IOC via Approximate Value Iteration (AVI) (32) has been proposed to address the inefficiency of calculating the log-partition (softened) value function in Equation 2.13. The main idea is that by employing AVI and estimating the expectation via Monte Carlo sampling the high complexity of calculating the log-partition function can be reduced.

The problem is that the added complexity of performing Monte Carlo sampling negates a large portion of the speedup generated via Approximate Value Iteration (50). This thesis outlines an analogous method for approximating the partition function in Maximum Entropy

IOC without the need for value iteration or Monte Carlo sampling resulting in a more efficient approach while preserving bounded error approximation guarantees.

2.7 Maximum Margin Planning

Maximum Margin Planning (63) addresses the problem of learning a complex cost function that motivates demonstrated behavior by reducing imitation learning to a maximum margin classification problem. This is done by utilizing the hinge-loss to form the following convex objective function,

$$C_q(\theta) = \frac{1}{n}\sum_{i=1}^{n} \beta_i \Big(\max_{\hat{\mu}\in\mathcal{G}_i} (\theta^T \mathbf{F}_i + l_i^T)\hat{\mu} - \theta^T \mathbf{F}_i \tilde{\mu}_i \Big)^q + \frac{\lambda}{2}\|\theta\|^2, \tag{2.17}$$

here $\hat{\mu}$ are expected state exploration counts under the Markov Decision Process $\mathcal{G}$, $\tilde{\mu}_i$ are observed exploration counts for trajectory i, and l_i is the state loss, $l_i : \mathcal{S} \to \mathbb{R}$, that penalizes states explored in trajectory i.

The method detailed in Algorithm 2 functions by optimizing the parameter vector θ so that the set of cost-augmented demonstrated trajectories are optimal.

Algorithm 2 Maximim Margin Planning

Input: State features $\mathbf{f}$ and empirical feature expectations $\tilde{\mathbb{E}}\Big[\sum_{j=1} \mathbf{f}\Big]_i$ for each cost map i

Output: Parameter vector θ

Set $\theta = 0$

for *t = 1:T* **do**

Compute π^* and $\mathbb{E}\Big[\sum_{j=1} \mathbf{f}(s_i)|\pi^*\Big]_i$ for each loss-augmented cost map $(\theta^T \mathbf{f}_i + l_i^T)$.

Set $\theta = \frac{1}{t}\Big[(t-r)\theta - rC\sum_{i=1}\Big(\mathbb{E}\Big[\sum_{j=1} \mathbf{f}|\pi^*\Big]_i - \tilde{\mathbb{E}}\Big[\sum_{j=1} \mathbf{f}\Big]_i\Big)\Big]$.

end

return θ

While this approach finds a unique solution under ideal settings and requires an optimal control solution as a sub-routine rather than integrating over the set of possible sequences, it assumes the demonstrated behavior is optimal and can be explained by a single cost function which is not always the case in complex tasks. Maximum Entropy IOC (82) addresses this issue by maximizing the causal entropy of the policy distribution allowing for sub-optimal and non-uniform behavior in the demonstration set.

2.8 Continuous Maximum Entropy Inverse Optimal Control

Maximum Entropy IOC in continuous settings is difficult in general because distribution normalization and expectation calculations require integrating over state and action variables

where the $\mathrm{softmax}$ function in Equation 2.13 is a smoothed interpolation of the maximum function,

$$\underset{\mathbf{a}\in\mathbf{A}_{\mathbf{s}_t}}{\mathrm{softmax}}\, \mathrm{Q}(\mathbf{s}_t, \mathbf{a}_t) = \log \int_{\chi} e^{\mathrm{Q}(\mathbf{s}_t, \mathbf{a}_t)} \mathrm{d}a_t. \tag{2.18}$$

The recursion in Equation 2.4, Equation 2.13 and Equation 2.14 can be analytically solved for continuous state-action representations of high-dimensional settings when the dynamics are linear (78; 77; 46),

$$\vec{s}_{t+1} = \mathbf{A}\vec{s}_t + \mathbf{B}\vec{a}_t, \tag{2.19}$$

and the features are quadratic, with parameter matrix $\mathbf{M}$,

$$\mathrm{cost}(\vec{s}_{1:T}) = \mathrm{M} \cdot \textstyle\sum_{t=1}^{T} \vec{s}_t \vec{s}_t^{T} = \sum_{t=1}^{T} \vec{s}_t^{T} \mathrm{M} \vec{s}_t. \tag{2.20}$$

Under these assumptions, the normalization factors for the distribution over trajectories ((Equation 2.13)) can be obtained in closed form and correspond to a conditional Gaussian distribution for the control policies (78; 77; 46; 52). This linear-quadratic assumption for IOC has been employed in linear-quadratic regulation settings (77; 52), and for non-linear-quadratic systems by locally making linear and quadratic approximations (46).

2.9 Locally Optimal Continuous Inverse Optimal Control

Locally optimal continuous inverse optimal control (46) addresses the complexity problem of calculating the partition function by locally approximating the probability distribution using local cost approximation.

In the continuous setting, Equation 2.12 can be expressed as,

$$\mathrm{P}(\mathbf{a}|\mathbf{s}_0) = e^{-\mathrm{cost}(\mathbf{a})} \int e^{-\mathrm{cost}(\tilde{\mathbf{a}})} \mathrm{d}\tilde{\mathbf{a}}. \tag{2.21}$$

where $\mathrm{cost}(\mathbf{a})$ is the expected cumulative cost of the path from state $\mathbf{s}_0$. A second order Taylor expansion of the cost around $\mathbf{a}$ can then be used to approximate the log likelihood function without computing the partition function,

$$\mathcal{L} = \frac{1}{2}(\mathbf{g}^{\mathsf{T}}\mathbf{H}^{-1}\mathbf{g} + \log|-\mathbf{H}|) - \frac{\mathrm{d}_{\mathbf{a}}}{2}\log 2\pi, \tag{2.22}$$

with gradient $\mathbf{g}$ and Hessian $\mathbf{H}$.

Unfortunately this method only works for deterministic fixed horizon problems and assumes cost functions are solely locally optimal ignoring any useful information from globally optimality when it exists. Additionally, while the method can be applied for inverse linear quadratic regulation (LQR) tasks, its deterministic assumption does not allow for the inclusion of Gaussian noise reducing the space of problems to which it can be applied. The predictive and performance guarantees of inverse optimal control also only apply within the local neighborhood

of the trajectory under the approximation assumptions making the resulting model difficult to apply in new situations in which an appropriate reference trajectory is not known.

2.10 Maximum Entropy Modeling via Symmetric Partition Functions

Symmetric partition functions have been used to address the intractability of using maximum entropy methods to form probabilistic models of continuous path distributions in high-dimensional spaces with features that possess low-dimensional structures (75).

This can be done without inefficient sampling based inference methods and without assuming the demonstrated sequences lie near a low dimensional sub-manifold, as in standard dimensionality-reduction methods.

The key assumption of this method is that if we can express features as integrals of feature potentials ψ_j over trajectories ξ,

$$\mathbf{f}_j(x) = \int_0^T \psi_j(\xi) ds, \tag{2.23}$$

then ψ_j can be compressed into a lower dimensional subspace without losing information,

$$\psi_j(s) = \psi_j(WW^T s), \forall j, s, \tag{2.24}$$

for an N-dimensional state $s \in \mathbb{R}^N$ and some given Nxd matrix W where $d < N$. This implies that the corresponding partition function is compressible as well which can then be calculated more efficiently via dynamic programming than the original, higher-dimensional, form.

Unfortunately this approach requires that the feature potentials are derived from states in $\mathbb{R}^N$ and can be compressed losslessly in order to efficiently calculate the partition function which is not guaranteed in decision problems with complex features.

Part I

Approximating Path Distributions in Weighted Graphs

Partially published in the Proceedings of the Neural Information Processing Systems Conference (https://papers.nips.cc/paper/5889-softstar-heuristic-guided-probabilistic-inference) (50).

Approximation approaches to probabilistic structured prediction include approximate max-ent IOC (32) and graph-based IOC (12). However, these methods require exploring large areas of sub-optimal paths and provide few guarantees on the accuracy of the approximations; they are not complete and the set of variable assignments uncovered may not be representative of the model's distribution.

Probabilistic inference models (72; 26; 71) estimate plan distributions that can be used to update a learned cost/reward function to more accurately match the feature preferences of observed behavior (15; 45). Unfortunately, many of the these probabilistic inference models fail to scale well to large decision processes and again provide few approximation guarantees.

Seeking to provide stronger guarantees and improve efficiency over previous methods, we present a heuristic-guided probabilistic search framework for estimating path near-optimal distributions in weighted graphs. Our approach generalizes the A* search algorithm (18) to calculate distributions over decision sequences through a state- space graph. This distribution can then be used to update a set of trainable parameters, θ, that motivate the behavior of the decision process via a cost/reward function (54; 1; 5; 81).

In the next chapter we first present an algorithm for heuristic-guided softened value iteration that enables the maximum entropy inverse optimal control framework to be more efficiently scaled to large decision processes by reducing the exploration of sub-optimal paths to the goal.

Despite the improved computational performance of this method in practice, as a sampling based approach it does not provide strong approximation guarantees. To address this issue we then present *SoftStar*, a heuristic-guided probabilistic search algorithm for inverse optimal control. This approach generalizes the A* search algorithm (18) to calculate distributions over decision sequences in predictive IOC settings allowing for efficient bounded approximations of the near-optimal path distribution through a decision space. We establish and analyze the theoretical guarantees of this approach and discuss the analogies to A* search.

We then demonstrate the effectiveness of the proposed algorithms in two settings: learning a predictive model for planning stroke trajectories for Latin characters and modeling the ball-handling decision process of professional soccer.

The first task of planning a sequence of pen strokes that produce a written character (41; 42) requires a state-space that includes remembering which of the previous line segments of the character have already been drawn—a power-set of the number of line segments. When re-tracing of line segments and pen lifts are allowed (both common in demonstrations), the set of trajectories that successfully complete the character grows infinitely. Yet, human demonstrations of writing trajectories in these tasks are typically within a very small portion of this feasible set of trajectories and are characterized by specific measurable tendencies (e.g. smooth transition angles between consecutively drawn line segments).

We also examine the task of spatially modeling the ball-handling decision processes of professional soccer players in single possession plays from given tracking data. There has been a lot of recent work in the area of modeling the outcomes of sports matches (48; 7; 35).

However, there has been little work on modeling the actual decision process of a professional soccer player on an action-by-action basis. This may be in large part due to the lack of needed tracking data on the player positions and the difficulty of framing the problem in an appropriate manner. When considering the types of actions that a player may take (pass, shot, clear, dribble, or cross) and the possible destination of the ball given that action (anywhere on the field for most of the actions), we are left with a very large action space and sequential decision process. However, we hypothesize that learning the common characteristics of the behavior of professional soccer players should reduce the feasible space of these decisions to a much smaller set that those players actually employ.

CHAPTER 3

HEURISTIC-GUIDED SOFTENED VALUE ITERATION

Inspired by techniques for efficient planning and heuristic-guided (A*) search (29), this chapter introduces a heuristic-guided value iteration algorithm for maximum entropy inverse optimal control that efficiently approximates near-optimal path distributions through large state-space graphs.

Standard value iteration methods begin without knowledge of the possible values of the neighboring states. This leads to the exploration of large areas of sub-optimal paths to the goal. By incorporating an admissible heuristic as an estimate for the neighboring state values that have not been calculated, we can greatly reduce the area of the state-space that needs to be explored before converging to an appropriate estimation of the state values and the resulting policy. This allows for us to efficiently perform value iteration in large decision processes.

In this chapter we develop a method for heuristic-guided value iteration that significantly improves the efficiency of standard value iteration when an admissible heuristic can be formed. This enables the maximum entropy inverse optimal control framework to be more efficiently scaled to large decision processes by reducing the exploration of sub-optimal paths to the goal leading to faster convergence on the near-optimal path distribution. We apply this method to the inverse optimal control problems of learning human behavior preferences in writing latin characters and the ball handler decision process in professional soccer. We refer to Chapter 5 for more detail on the results of these experiments.

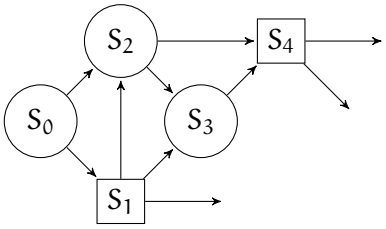

Figure 1: State expansion map. Rectangular nodes are members of the approximation set $\mathcal{S}_{\text{approx}}$. Though they lead to other states, a heuristic function, $V^{+}(s)$, is employed to approximate their values rather than recursively computing them.

3.1 Heuristic-guided policy approximation

We begin by reducing the space of the decision process by truncating the softmax value iteration recurrence of Equation 2.13 at different states $s \in \mathcal{S}_{\text{approx}}$, as shown in Figure 1 (for notational convenience, we employ time-invariant value functions). At those states, the exact softmax value function $V_{\text{soft}}(s)$ that would need to be computed, is instead replaced by a heuristic value, the upper bound $V^{+}_{\text{soft}}(s)$ ($V_{\text{soft}}(s) \leq V^{+}_{\text{soft}}(s)$). We assume monotonic heuristic functions for estimating the values of unexplored states,

$$V^{+}_{\text{soft}}(s) \geq \operatorname*{softmax}_{a} \mathbb{E}[V^{+}_{\text{soft}}(s_t)|a, s] + \text{reward}_{\theta}(a, s)]$$

—i.e., subsequent heuristics are not looser bounds.

We could choose the approximate state set and directly employ this approximate policy for rejection sampling or importance sampling. Sampling-based approaches of this sort have been recently investigated (3; 10). However, since the true solution policy π_{soft} is unknown,

this may be extremely sample-inefficient. Instead, we propose to iteratively refine our policy estimate.

3.2 Greedy selection of the approximation set

A natural approach for choosing set $\mathcal{S}_{\text{approx}}$ is to sequentially expand the most potentially beneficial state. If rewards are unbounded (i.e., can be $-\infty$), this suggests the following optimization:

$$\underset{s_q \in s_{1:T}}{\operatorname{argmax}} \mathbb{E}_{\pi^+\mathcal{T}}[V(s_q)^+]. \tag{3.1}$$

This algorithm is very similar to A* search and fast convergence to the true distribution is easily shown in finite state settings. However, one key difference is that the priority for expanding a state s_q depends on the $\operatorname{softmax}$ of all paths to s_q, which is equivalent to the expected occurrence of state s_q in the approximate sequence distribution, $\mathbb{E}_{\pi^+\mathcal{T}}[s_q \in s_{1:T}]$. Unfortunately, this small difference has large computational implications. In optimal path planning with an admissible monotonic heuristic, the minimum cost path to a state does not change. However, since $\mathbb{E}_{\pi^+\mathcal{T}}[V(s_q)^+]$ depends on all paths to state s_q, all paths created by modifying $\mathcal{S}_{\text{approx}}$ must be considered.

3.3 Heuristic-Guided Softened Value Iteration

To avoid the expensive computations needed for greedily constructing the approximation set, we instead randomize. The following algorithm iteratively improves our softmax policy estimate, π_{soft}, by selectively refining the set of approximated states $\mathcal{S}_{\text{approx}}$ using trajectory

samples. The algorithm samples these trajectories according to the current policy estimate, $a_t|s_t \sim \pi^+_{\text{soft}}(a_t|s_t) \triangleq \text{sample}(s_t, \pi^+_{\text{soft}})$ and transition function $\mathcal{T}(s_{t+1}|a_t, s_t)$. When an approximated state, $s \in \mathcal{S}_{\text{approx}}$, is encountered, it is removed from the approximation set and its possible subsequent states are added to the set. This continues until the trajectory either reaches a goal state or a maximum allowable length. The values along this sampled trajectory are then updated using softened maximum entropy value iteration via (Equation 2.13). The values and the policy of each state in the sequence are updated in reverse allowing for the updated policy of each state to take into account the value of each potential future state in the sequence.

Consider the state map depicted in Figure 1. Here, states S_0, S_2, and S_3 have been fully expanded and have updated state values according to (Equation 2.13). States S_1 and S_4 are members of $\mathcal{S}_{\text{approx}}$ as they have not been fully expanded and their state values are approximated using an admissible heuristic value V^+_{soft}. The algorithm expands the states that have a higher probability of having large approximation loss. Once a state is expanded, its value is updated according to (Equation 2.13) and its policy is updated according to (Equation 2.14).

Algorithm 3 Heuristic-Guided Stochastic Softened Value Iteration

Input: Heuristic function V^+

Output: A policy estimate, π^+_{soft}, and state value estimates, V

Set $V = V^+$

Calculate π^+_{soft} via (Equation 2.14)

while V *not converged* **do**

clear sequence

add initial state to sequence

while $s_t \neq s_{goal}$ *and* $|sequence| \neq maxHorizon$ **do**

$a_t = sample(s_t, \pi^+_{\text{soft}})$ if precomputed or via (Equation 2.14)

$s_t = \mathcal{T}(s_t, a_t)$

add s_t to sequence.

end

for *i = (sequence size-1) : 1* **do**

update $V(s_i)$ via (Equation 2.13)

update $\pi_{\text{soft}}(s_i|a_t)$ via (Equation 2.14)

end

end

It is worthwhile to note that while the heuristic function is problem specific, the inference algorithm functions with no knowledge of the problem domain and is completely generalizable graph-based decision problems.

CHAPTER 4

SOFTSTAR: BOUNDED APPROXIMATE PATH DISTRIBUTIONS VIA HEURISTIC-GUIDED SEARCH

Partially published in the Proceedings of the Neural Information Processing Systems Conference (https://papers.nips.cc/paper/5889-softstar-heuristic-guided-probabilistic-inference) (50).

The problem with the sampling based approach presented in the previous chapter is that there is a lack of bounded approximation and complexity guarantees. Symmetries in the decision process can be exploited to improve efficiency (75), but decision processes are not guaranteed to be (close to) symmetric. Other approaches to approximate probabilistic structured prediction include approximate maxent IOC (32), heuristic-guided sampling (Algorithm 3 (51)), and beam search (40; 72; 71) which considers a bounded-width best-first search over possible variable assignments (47; 17; 24; 59). However, few guarantees are provided by these approach; they are not complete and the set of variable assignments uncovered may not be representative of the problems true distribution.

Seeking to provide stronger guarantees and more efficiency than previous methods, we present *SoftStar*, a heuristic-guided probabilistic search algorithm for predictive inverse optimal control. Our approach parallels the A* search algorithm (18) to calculate distributions over decision sequences in predictive inverse optimal control settings allowing for an efficient bounded approximation of the near-optimal path distribution through a decision space. We establish and analyze the theoretical guarantees of this approach, discuss the analogies to A*

search, and demonstrate its effectiveness in three settings: a synthetic task, learning a predictive model for planning stroke trajectories for Latin characters, and modeling the ball-handling decision process of professional soccer.

Motivated by the efficiency improvements of heuristic-guided search algorithms for optimal planning, we define an analogous approximation task in the predictive inference setting and present algorithms that leverage heuristic functions to accomplish this task efficiently with bounded approximation error.

In this section we consider an action cost rather than a reward such that maximum entropy IOC algorithms (81; 78) estimate a stochastic action policy that is most uncertain while still guaranteeing the same expected cost as demonstrated behavior on an unknown cost function (1). For planning settings, this formulation yields a probability distribution over state sequences that are consistent with paths through the state-space graph, $\hat{\mathrm{P}}(\mathbf{s}_{1:\mathrm{T}}) \propto e^{-\mathrm{cost}_\theta(\mathbf{s}_{1:\mathrm{T}})}$, where $\mathrm{cost}_\theta(\mathbf{s}_{1:\mathrm{T}}) = \sum_{t=1}^{T-1} \theta^{\mathrm{T}} \mathbf{f}(s_t, s_{t+1})$ is a linearly weighted vector of state-transition features combined using the feature function, $\mathbf{f}(\mathbf{s}_t, \mathbf{s}_{t+1})$, and a learned parameter vector, θ. Calculating the marginal state probabilities of this distribution is important for estimating model parameters. The forward-backward algorithm (6) can be employed, but for large state-spaces it may not be practical.

4.1 Inference as softened planning

We begin our investigation for finding a more efficient inference algorithm to support prediction and learning in predictive inverse optimal control models by recasting the inference task from the perspective of softened planning.

We define the desired next state at time-step t as a_t. This is the decision/action made at time-step t and is identical to the next state in deterministic settings. In this setting the probability of transitioning to state s_{t+1} from state s_t when taking action a_t is $p(s_{t+1}|s_t, a_t)$ given the transition mapping $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S}' \rightarrow \mathcal{P}(\mathcal{S}'|\mathcal{S}, \mathcal{A})$. The cost of a decision can then be represented via a linear combination of state-action features and the set of learned parameters, $cost(s_t, a_t) = \theta^T \mathbf{f}(s_t, a_t)$, such that the predictive inverse optimal control distribution over state sequences factors into a stochastic policy (81),

$$\pi(a_t|s_t) = e^{E_\tau[h_{soft}(s_{t+1})|s_t, a_t] - h_{soft}(s_t) - \theta^T \mathbf{f}(s_t, a_t)}, \tag{4.1}$$

according to a softened cost-to-go[1], $h_{soft}(s)$, recurrence relationship that is a relaxation of the Bellman equation:

$$\begin{aligned} h_{soft}(s_t) &= \underset{a_t \in \mathcal{A}(s_t)}{\text{softmin}} \left\{ \sum_{s_{t+1} \in \mathcal{N}(s_t)} p(s_{t+1}|s_t, a_t) h_{soft}(s_{t+1}) + \theta^T \mathbf{f}(s_t, a_t) \right\} \\ &= \underset{a_t \in \mathcal{A}(s_t)}{\text{softmin}} \left\{ E_\tau[h_{soft}(s_{t+1})|s_t, a_t] + \theta^T \mathbf{f}(s_t, a_t) \right\}, \end{aligned} \tag{4.2}$$

taking into account the expected cost-to-go of the next state, $E_\tau[h_{soft}(s_{t+1})|s_t, a_t]$, given the transition probabilities, $\mathcal{T}$, where Ξ_{s_t, s_T} is the set of all paths from s_t to s_T; the softmin, $\underset{x}{\text{softmin}}\, \alpha(x) \triangleq$

[1] We assume a time-invariant softened cost-to-go function in our formulation for notational convenience. Time-varying formulations are also easily obtained.

$-\log \sum_x e^{-\alpha(x)}$, is a smoothed relaxation of the $\min$ function[1]; $p(s_{t+1}|s_t, a_t)$ is the probability of transitioning into state s_{t+1} from state s_t when taking action a_t; and the terminal state value is initially 0 for the goal state ($-\infty$ for others).

A similar softened minimum distance exists in the forward direction from the start state. Here we consider the probability that each action will transition state s_{t-1} to state s_t in order to calculate the expected cost of the transition, $E_\tau[\theta^T \mathbf{f}(s_{t-1}, a_{t-1})|s_t, s_{t-1}]$:

$$\begin{aligned} d_{\text{soft}}(s_t) &= \underset{s_{t-1}\in\mathcal{N}(s_t)}{\text{softmin}} \left\{ d_{\text{soft}}(s_{t-1}) + \sum_{a_{t-1}\in\mathcal{A}(s_{t-1})} p(s_t|s_{t-1}, a_{t-1})\theta^T \mathbf{f}(s_{t-1}, a_{t-1}) \right\} \\ &= \underset{s_{t-1}\in\mathcal{N}(s_t)}{\text{softmin}} \left\{ d_{\text{soft}}(s_{t-1}) + E_\tau[\theta^T \mathbf{f}(s_{t-1}, a_{t-1})|s_t, s_{t-1}] \right\}. \end{aligned} \tag{4.3}$$

By combining forward and backward soft values, important marginal probabilities and expectations can be obtained that are needed to predict state visitation probabilities and fit the maximum entropy inverse optimal control model's parameters (81). For example, the probability of the transition from s_a to s_b under the soft path distribution is:

$$e^{-d_{\text{soft}}(s_a) - E_\tau[h_{\text{soft}}(s_b)|s_a,a] - \theta^T \mathbf{f}(s_a,a) + d_{\text{soft}}(s_T)}. \tag{4.4}$$

Thus, efficient search and learning require accurate estimates of d_{soft} and h_{soft} values.

Likewise, combining the softened minimum distance, d_{soft}, with an admissible (lower bound) heuristic for estimating the expected softened cost-to-go, $\hat{h}_{\text{soft}}$, allows us to estimate the ex-

[1] Equivalently, $\min_x \alpha(x) + \underset{x}{\text{softmin}} \left\{ \alpha(x) - \min_x \alpha(x) \right\}$ is employed to avoid overflow/underflow in practice.

pected cumulative cost of a trajectory from the initial state to the goal that travels through state s_t,

$$f_{\text{soft}}(s_t) = d_{\text{soft}}(s_t) + \hat{h}_{\text{soft}}(s_t), \tag{4.5}$$

which is essential for utilizing the A* framework of heuristic-guided search outlined in Section 2.3.

The softened cost-to-go and distance functions for deterministic graphs can be computed in closed-form using a geometric matrix series (with actions represented by succeeding states):

$$\mathbf{A} = \begin{bmatrix} e^{-\text{cost}(s_1,s_1)} & e^{-\text{cost}(s_1,s_2)} & \dots & e^{-\text{cost}(s_1,s_n)} \\ e^{-\text{cost}(s_2,s_1)} & e^{-\text{cost}(s_2,s_2)} & \dots & e^{-\text{cost}(s_2,s_n)} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-\text{cost}(s_n,s_1)} & e^{-\text{cost}(s_n,s_2)} & \dots & e^{-\text{cost}(s_n,s_n)} \end{bmatrix}$$

$$\mathbf{B} = \mathbf{A}(\mathbf{I} - \mathbf{A})^{-1} = \mathbf{A} + \mathbf{A}^2 + \mathbf{A}^3 + \mathbf{A}^4 + \cdots . \tag{4.6}$$

The $(i,j)^{\text{th}}$ entry of $\mathbf{B}$ is related to the softmin of all the paths from s_i to s_j. Specifically, the softened cost-to-go can be written as $h_{\text{soft}}(s_i) = -\log b_{s_i,s_T}$. Unfortunately, the required matrix inversion operation is computationally expensive, preventing its use in typical inverse optimal control applications. In fact, power iteration methods used for sparse matrix inversion closely

resemble the softened Bellman updates of (Equation 4.2) that have instead been used for IOC (78).

4.2 Challenges and approximation desiderata

In contrast with optimal control and planning tasks, softened distance functions, $\mathrm{d}_{\text{soft}}(\mathrm{s})$, and cost-to-go functions, $\mathrm{h}_{\text{soft}}(\mathrm{s})$, in predictive inverse optimal control are based on many paths rather than a single (best) one. Thus, unlike in A* search, each sub-optimal path cannot simply be ignored; its influence must instead be incorporated into the softened distance calculation (Equation 4.2). This key distinction poses a significantly different objective for heuristic-guided probabilistic search:

Find a subset of paths for which the softmin distances closely approximate the softmin of the entire path set.

While we would hope that a small subset of paths exists that provides a close approximation, the cost function weights and the structure of the state-space graph ultimately determine if this is the case. With this in mind, the desiderata for an algorithm that seeks a small approximation set are that it provides:

1. *Known bounds on approximation guarantees;*
2. *Convergence to any desired approximation guarantee;*
3. *Efficient finding small approximation sets of paths.*

We construct algorithms with these considerations in mind and analyze their guarantees and behaviors.

4.3 Regimes of Convergence

In A* search, theoretical results are based on the assumption that all infinite length paths have infinite cost (i.e., any cycle has a positive cost) (29). This avoids a negative cost cycle regime of non-convergence. Leading to a stronger requirement for our predictive setting are three regimes of convergence for the predictive inverse optimal control distribution, characterized by:

1. *An infinite-length most likely plan;*
2. *A finite-length most likely plan with expected infinite-length plans; and*
3. *A finite expected plan length.*

The first regime results from the same situation described for optimal planning: reachable cycles of negative cost. The second regime arises when the number of paths grows faster than the penalization of the weights from the additional cost of longer paths (without negative cycles) and is non-convergent. The final regime is convergent. When inverse optimal control is based on finite-length demonstrations and sufficiently expressive feature representations (e.g., including a constant-value feature), the IOC cost estimate will correspond to this regime.

An additional assumption is needed in the predictive IOC setting to avoid the second regime of non-convergence. We assume that a fixed bound on the entropy of the distribution of paths, $\mathrm{H}(\mathbf{s}_{1:\mathrm{T}}) \triangleq \mathbb{E}[-\log \mathrm{P}(\mathbf{s}_{1:\mathrm{T}})] \leq \mathrm{H}_{\mathsf{max}}$, is known.

Theorem 1. *Expected costs under the predictive IOC distribution are related to entropy and softmin path costs by:*

$$\mathbb{E}[\mathit{cost}_\theta(\mathbf{s}_{1:T})] = \mathrm{H}(\mathbf{s}_{1:T}) - \mathrm{d}_{\mathit{soft}}(s_T).$$

Proof. By writing the definition of the entropy, we have:

$$\mathrm{H}(\mathbf{s}_{1:T}) = \mathbb{E}\left[-\log \mathrm{P}(\mathbf{s}_{1:T})\right] = \mathbb{E}\left[-\log \frac{e^{-\mathsf{cost}_\theta(\mathbf{s}_{1:T})}}{\sum_{\mathbf{s}_{1:T}} e^{-\mathsf{cost}_\theta(\mathbf{s}_{1:T})}}\right] = \mathbb{E}\left[\mathsf{cost}_\theta(\mathbf{s}_{1:T})\right] - \mathrm{d}_{\mathsf{soft}}(s_{\mathsf{goal}}).$$

□

Together, bounds on the entropy and softmin distance function constrain expected costs under the predictive IOC distribution (Theorem 1).

4.4 Computing approximation error bounds

A* search with a non-monotonic heuristic function guarantees optimality when the priority queue's minimal element has an estimate $\mathrm{d}_{\mathsf{soft}}(\mathrm{s}) + \hat{\mathrm{h}}_{\mathsf{soft}}(\mathrm{s})$ exceeding the best start-to-goal path cost, $\mathrm{d}_{\mathsf{soft}}(s_T)$. Though optimality is no longer guaranteed in the softmin search setting, approximations to the softmin distance are obtained by considering a subset of paths (Lemma 1).

Lemma 1. *Let Ξ represent the entire set (potentially infinite in size) of paths from state s to s_T. We can partition the set Ξ into two sets Ξ_a and Ξ_b such that $\Xi_a \cup \Xi_b = \Xi$ and $\Xi_a \cap \Xi_b = \emptyset$ and*

define d_{soft}^{Ξ} *as the* $\mathrm{softmin}$ *over all paths in set* Ξ*. Then, given a lower bound estimate for the distance,* $\hat{\mathrm{d}}_{soft}(\mathrm{s}) \leq \mathrm{d}_{soft}(\mathrm{s})$*:*

$$e^{-\mathrm{d}_{soft}^{\Xi}(\mathrm{s})} - e^{-\mathrm{d}_{soft}^{\Xi_a}(\mathrm{s})} \leq e^{-\hat{\mathrm{d}}_{soft}^{\Xi_b}(\mathrm{s})}. \tag{4.7}$$

Proof. By definition,

$$\sum_{\text{path}\in\Xi} e^{-\text{cost(path)}} = \sum_{\text{path}\in\Xi_a} e^{-\text{cost(path)}} + \sum_{\text{path}\in\Xi_b} e^{-\text{cost(path)}} \leq \sum_{\text{path}\in\Xi_a} e^{-\text{cost(path)}} + \sum_{\text{path}\in\Xi_b} e^{-\widehat{\text{cost}}(\text{path})}$$

for $\widehat{\mathrm{cost}}(\mathrm{path}) \leq \mathrm{cost}(\mathrm{path})$. Equivalently, for $\hat{\mathrm{d}}_{\text{soft}}(\mathrm{s}) \leq \mathrm{d}_{\text{soft}}(\mathrm{s})$: $e^{-\mathrm{d}_{\text{soft}}^{\Xi}(\mathrm{s})} - e^{-\mathrm{d}_{\text{soft}}^{\Xi_a}(\mathrm{s})} \leq e^{-\hat{\mathrm{d}}_{\text{soft}}^{\Xi_b}(\mathrm{s})}$.

□

We establish a bound on the error introduced by considering the set of paths through a set of states $\mathcal{S}_{\approx}$ in the following Theorem.

Theorem 2. *Given an approximation state subset* $\mathcal{S}_{\approx} \subseteq \mathcal{S}$ *with neighbors of the approximation set defined as* $\mathcal{N}(\mathcal{S}_{\approx}) \triangleq \bigcup_{\mathrm{s}\in\mathcal{S}_{\approx}} \mathcal{N}(\mathrm{s}) - \mathcal{S}_{\approx}$*, the approximation loss of exact search for paths through this approximation set (i.e., paths with non-terminal vertices from* $\mathcal{S}_{\approx}$ *and terminal vertices from* $\mathcal{S}_{\approx} \cup \mathcal{N}(\mathcal{S}_{\approx})$*) is bounded by the softmin of the set's neighbor's estimates:*

$$e^{-\mathrm{d}_{soft}(\mathrm{s}_T)} - e^{-\mathrm{d}_{soft}^{\mathcal{S}_{\approx}}(\mathrm{s}_T)} \leq e^{-\,\mathrm{softmin}_{\mathrm{s}\in\mathcal{N}(\mathcal{S}_{\approx})}\left\{\mathrm{d}_{soft}^{\mathcal{S}_{\approx}}(\mathrm{s}) + \hat{\mathrm{h}}_{soft}(\mathrm{s})\right\}}, \tag{4.8}$$

where $\mathrm{d}_{soft}^{\mathcal{S}_{\approx}}(\mathrm{s})$ *is the softmin of all paths with terminal state* s *and all previous states within* $\mathcal{S}^{\approx}$*.*

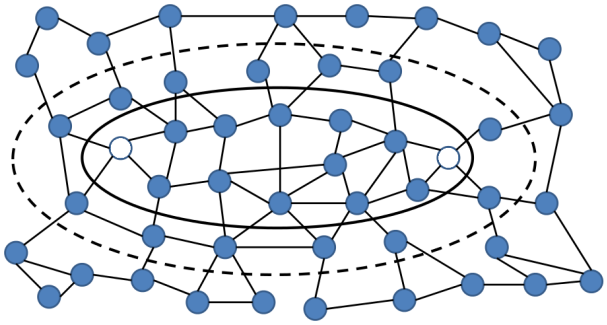

Figure 1: The approximation state set (within solid line) and the approximation neighbor set (between solid line and dashed line) with start and goal states (non-shaded nodes).

Proof. Theorem 2 follows directly from Lemma 1 by choosing the set of paths in the approximation set for Ξ_a and the set of paths not contained within the approximation set for Ξ_b. The costs of Ξ_b are lower bounds estimated by adding the cost within the approximation set to the heuristic found at the path's first state outside of the approximation set. □

Thus, for a dynamic construction of the approximation set $\mathcal{S}^{\approx}$, a bound on approximation error can be maintained by tracking the weights of all states in the neighborhood of that set (Figure 1). In practice, even computing the exact softened distance function for paths through a small subset of states may be computationally impractical. Theorem 3 establishes the approximate search bounds when only a subset of paths in the approximation set are employed to compute the soft distance.

Theorem 3. *If a subset of paths $\Xi'_{\mathcal{S}_\approx} \subseteq \Xi_{\mathcal{S}_\approx}$ (and $\bar{\Xi}'_{\mathcal{S}_\approx} \subseteq \Xi_{\mathcal{S}_\approx} - \Xi'_{\mathcal{S}_\approx}$ represents a set of paths that are prefixes for all of the remaining paths within $\mathcal{S}_\approx$) through the approximation set $\mathcal{S}_\approx$ is employed to compute the soft distance, the error of the resulting estimate is bounded by:*

$$
\begin{aligned}
& e^{-\mathrm{d}_{soft}(\mathrm{s}_T)} - e^{-\mathrm{d}_{soft}^{\Xi'_{\mathcal{S}_\approx}}(\mathrm{s}_T)} \leq \\
& \exp\left\{ -\operatorname{softmin}\left(\operatorname*{softmin}_{\mathrm{s}\in\mathcal{N}(\mathcal{S}_\approx)} \left\{ \mathrm{d}_{soft}^{\Xi'_{\mathcal{S}_\approx}}(\mathrm{s}) + \hat{\mathfrak{h}}_{soft}(\mathrm{s}) \right\}, \right.\right. \\
& \qquad \left.\left. \operatorname*{softmin}_{\mathrm{s}\in\mathcal{S}_\approx} \left\{ \mathrm{d}_{soft}^{\bar{\Xi}'_{\mathcal{S}_\approx}}(\mathrm{s}) + \hat{\mathfrak{h}}_{soft}(\mathrm{s}) \right\} \right) \right\}.
\end{aligned}
$$

Proof. Following directly from Lemma 1, the paths can be partitioned into the subset $\Xi'_{\mathcal{S}_\approx}$ and the subset of all other paths. The set of all other paths either terminate at $\mathcal{S}_\approx$'s neighbor set or have prefixes represented by the set $\bar{\Xi}'_{\mathcal{S}_\approx}$. □

4.5 SoftStar: Greedy forward path exploration and backward cost-to-go estimation

Our algorithm greedily expands nodes by considering the state contributing the most to the approximation bound (Theorem 3). This is accomplished by extending A* search in the following algorithm.

Algorithm 4 SoftStar: Greedy forward and approximate backward search with fixed ordering

Input: State-space graph $\mathcal{G}$, initial state s_1, goal s_T, heuristic $\hat{h}_{soft}$, and approximation bound ϵ

Output: Approximate soft distance to goal $h^{\mathcal{S}_{\approx}}_{soft}$ and approximate soft distance from initial $d^{\mathcal{S}_{\approx}}_{soft}$

Set $h_{soft}(s) = d_{soft}(s) = f_{soft}(s) = \infty \; \forall \; s \in \mathcal{S}$, $h_{soft}(s_T) = 0$, $d_{soft}(s_1) = 0$ and $f_{soft}(s_1) = \hat{h}_{soft}(s_1)$

Insert $\langle s_1, f_{soft}(s_1) \rangle$ into priority queue P and initialize empty stack O

while $\underset{s \in P}{\text{softmin}}(d_{soft}(s)) + \epsilon \leq d_{soft}(s_T)$ **do**

Set $s \rightarrow$ min element popped from P

Push s onto O

for $s' \in \mathcal{N}(s)$ **do**

$d_{soft}(s') = \underset{s'' \in \mathcal{N}(s')}{\text{softmin}} \left\{ d_{soft}(s'') + E_\tau[\theta^T \mathbf{f}(s'', a) | s'', s'] \right\}$ (Equation 4.3)

$f_{soft}(s') = d_{soft}(s') + \hat{h}_{soft}(s'))$ (Equation 4.5)

(Re-)Insert $\langle s', f_{soft}(s') \rangle$ into P

end

end

while *O not empty* **do**

Set $s \rightarrow$ top element popped from O

$h_{soft}(s) = \underset{a \in \mathcal{A}(s)}{\text{softmin}} \left\{ E_\tau[h_{soft}(s') | s, a] + \theta^T \mathbf{f}(s, a) \right\}$ (Equation 4.2)

end

return h_{soft} and d_{soft}.

For insertions to the priority queue, if s' already exists in the queue, its estimate is updated for both d_{soft} and f_{soft} and it is resorted into the queue. Additionally, the softmin of all of the estimates of elements on the queue can be dynamically updated as elements are added and removed.

The queue contains some states that have never been explored and some that have. The former correspond to the neighbors of the approximation state set and the latter correspond to the search approximation error within the approximation state set (Theorem 3). The softmin over all elements of the priority queue thus provides a bound on the approximation error of the returned distance measure. The exploration order, O, is a stack containing the order that each state is explored/expanded.

A loop through the reverse of the node exploration ordering (stack O) generated by the forward search computes complementary backward cost-to-go values, h_{soft}. The expected number of occurrences of state transitions can then be calculated for the approximate distribution (Equation 4.4). The bound on the difference between the expected path cost of this approximate distribution and the actual distribution over the entire state set is established in Theorem 4.

It should be noted that the backward approximate search algorithm is far more efficient than greedy forward search as the space is reduced to the set of nodes expanded by the forward pass.

Theorem 4. *The cost expectation inaccuracy introduced by employing state set $\mathcal{S}_{\approx}$ is bounded by:* $|\mathbb{E}[\mathit{cost}_\theta(\mathbf{S}_{1:\mathrm{T}})] - \mathbb{E}_{\mathcal{S}_{\approx}}[\mathit{cost}_\theta(\mathbf{S}_{1:\mathrm{T}})]| \leq \quad e^{\mathrm{d}^{\mathcal{S}_{\approx}}_{\mathit{soft}}(s_T) - \mathit{softmin}(\mathrm{P})} \left| \mathbb{E}_{\mathrm{P}}[\mathit{cost}_\theta(\mathbf{S}_{1:\mathrm{T}})] - \mathbb{E}_{\mathcal{S}_{\approx}}[\mathit{cost}_\theta(\mathbf{S}_{1:\mathrm{T}})] \right|,$

where: $\mathbb{E}_{S_{\approx}}$ *is the expectation under the approximate state set produced by the algorithms;* $\mathrm{softmin(P)}$ *is the softmin of elements remaining on the priority queue after the first while loop of Algorithm 4; and* $\mathrm{E_P}$ *is the expectation over all paths not considered in the second while loop (i.e., remaining on the priority queue).* $\mathrm{E_P}$ *is unknown, but can be bounded using Theorem 1.*

For learning purposes, appropriate regularization can be incorporated into parameter estimation based on these bounds (23).

4.5.1 Increasing Efficiency in Single Goal Graphs via Bidirectional Search

In problems with a single goal state we can further increase the search efficiency of the *SoftStar* algorithm, and utilize modern multi core CPU architecture by parallelizing the forward and backward passes through the state-space graph via bidirectional search, algorithm 5.

Algorithm 5 Bidirectional SoftStar

Input: State-space graph $\mathcal{G}$, initial state s_1, goal s_T, heuristic $\hat{h}_{\text{soft}}$, and approximation bound ϵ

Output: Approximate soft distance to goal $h^{\mathcal{S}_{\approx}}_{\text{soft}}$ and approximate soft distance from initial $d^{\mathcal{S}_{\approx}}_{\text{soft}}$

Set $h_{\text{soft}}(s) = d_{\text{soft}}(s) = f_{\text{soft}}(s) = \infty \; \forall \; s \in \mathcal{S}$, $h_{\text{soft}}(s_T) = 0$, $d_{\text{soft}}(s_1) = 0$, $f_{\text{soft}}(s_1) = \hat{h}_{\text{soft}}(s_1)$ and $f_{\text{soft}}(s_T) = \hat{d}_{\text{soft}}(s_T)$

Forward Search:

while $\underset{s \in P_{\textit{forward}}}{\text{softmin}}(d_{\textit{soft}}(s)) + \epsilon \leq d_{\textit{soft}}(s_T)$ **do**

Set $s \rightarrow$ min element popped from P_{forward}

for $s' \in N_{\textit{forward}}(s)$ **do**

$d_{\text{soft}}(s') = \hat{d}_{\text{soft}}(s') = \underset{s'' \in \mathcal{N}(s')}{\text{softmin}} \left\{ d_{\text{soft}}(s'') + E_\tau[\theta^{\mathrm{T}} \mathbf{f}(s'', a) | s'', s'] \right\}$

$f_{\text{soft}}(s') = \hat{d}_{\text{soft}}(s') + \hat{h}_{\text{soft}}(s'))$

(Re-)Insert $\langle s', f_{\text{soft}}(s') \rangle$ into P_{forward}

end

end

Reverse Search:

while $\underset{s \in P_{\textit{reverse}}}{\text{softmin}}(h_{\textit{soft}}(s)) + \epsilon \leq h_{\textit{soft}}(s_1)$ **do**

Set $s \rightarrow$ min element popped from P_{reverse}

for $s' \in N_{\textit{reverse}}(s)$ **do**

$h_{\text{soft}}(s') = \hat{h}_{\text{soft}}(s') = \underset{a \in \mathcal{A}(s')}{\text{softmin}} \left\{ E_\tau[h_{\text{soft}}(s'') | s', a] + \theta^{\mathrm{T}} \mathbf{f}(s', a) \right\}$

$f_{\text{soft}}(s') = \hat{d}_{\text{soft}}(s') + \hat{h}_{\text{soft}}(s'))$

(Re-)Insert $\langle s', f_{\text{soft}}(s') \rangle$ into P_{reverse}

end

end

return h_{soft} and d_{soft}.

The key idea is that while performing a forward search from the initial state to the goal motivated by a heuristic for the softened cost-to-go, $\hat{\mathrm{h}}_{\text{soft}}$, we can also do a reverse search from the goal to the initial state motivated by a heuristic for the softened distance, $\hat{\mathrm{d}}_{\text{soft}}$. In the reverse case we would be updating h_{soft} in the same way that the forward pass updates d_{soft}. If either pass (forward/backward) adds a state to its respective queue that has previously been expanded by the other pass, then the updated value for either h_{soft} or $\mathrm{d}$soft will be used in place of the heuristic for calculating f_{soft}. This eliminates the need for the additional backward pass with fixed ordering through the explored stack O in algorithm 4. When run in parallel, this should elicit a strong speed up while reducing overhead by using updated values in place of the heuristics once the search directions overlap.

4.6 Completeness guarantee

The notion of monotonicity extends to the probabilistic setting, guaranteeing that the expansion of a state provides no looser bounds than the unexpanded state (Definition 1).

Definition 1. *A heuristic function $\hat{\mathrm{h}}_{soft}$ is monotonic iff*

$$\forall \mathrm{s} \in \mathcal{S}, \hat{\mathrm{h}}_{soft}(\mathrm{s}) \geq \underset{\mathrm{a} \in \mathcal{A}(\mathrm{s})}{\operatorname{softmin}} \left\{ \mathrm{E}_{\tau}[\hat{\mathrm{h}}_{soft}(\mathrm{s}')|\mathrm{s}, \mathrm{a}] + \theta^{\mathrm{T}} \mathbf{f}(\mathrm{s}, \mathrm{a}) \right\}.$$

Assuming this, the completeness of the proposed algorithm can be established (Theorem 5).

Theorem 5. *For monotonic heuristic functions and finite softmin distances, convergence to any level of softmin approximation is guaranteed by Algorithm 4.*

4.7 Inference Comparisons on Synthetic Data

In this section we demonstrate the effectiveness of the *SoftStar* algorithm on a synthetic dataset in which we can control the degree of heuristic sub-optimality. We generate a random full cost matrix with 20,000 state nodes and compute exact soft distance values via (Equation 4.6), requiring over 10 minutes of CPU time for the inversion of the deterministic matrix. We evaluate our approach using a heuristic function $\hat{h}_{soft}(s) = \alpha h_{soft}(s)$ for different values of $\alpha \in [0, 1]$.

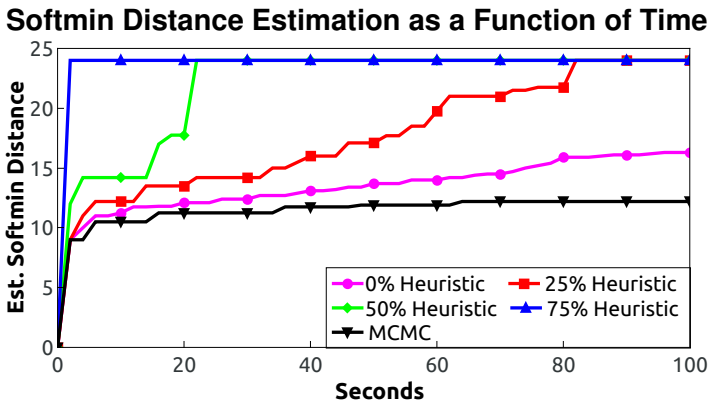

Figure 2: A comparison of inference efficiency using varying ratios of the true softmin distance as heuristic values.

Figure 2 shows the efficiency of our approach with heuristic functions of varying tightness. Without a heuristic ($\alpha = 0\%$), over 26 minutes of CPU time is required for the soft distance

estimate to converge, whereas for $\alpha \geq 75\%$ convergence occurs in about 1 second. MCMC performs poorly, getting caught in local optima and taking over 45 minutes of CPU time to converge.

4.8 Feature expectations and gradient computation

In order to generate the desired path distribution we must first run the SoftStar algorithm in order to calculate the softened cost-to-go values, d_{soft}, on the ordered set, O. We then use Equation 4.4 to compute the probability of each state transition in O as well as the expected feature distributions needed to generate the gradient:

$$\nabla L(\theta) = \mathbb{E}_{\hat{\pi}} \left[\sum_{t=0}^{\hat{T}-1} f(\mathbf{s}_t, \mathbf{a}_t) \right] - \mathbb{E}_{\tilde{\pi}} \left[\sum_{t=0}^{\tilde{T}-1} f(\mathbf{s}_t, \mathbf{a}_t) \right]. \tag{4.9}$$

CHAPTER 5

EXPERIMENTAL VALIDATION AND DISCUSSION

Previously published in the Proceedings of the Neural Information Processing Systems Conference (https://papers.nips.cc/paper/5889-softstar-heuristic-guided-probabilistic-inference) (50).

We demonstrate the effectiveness of our presented approaches on datasets for Latin character construction using sequences of pen strokes and ball-handling decisions of professional soccer players.

We employ stochastic accelerated gradient descent with an adagrad learning rate and accelerated momentum (21; 68) to estimate the cost parameters, θ. We refer to those papers for details of the methods.

5.1 Comparison approaches

We compare our approach with reasonable alternatives: heuristic guided maximum entropy sampling (Algorithm 3), approximate maximum entropy sampling (32), reversible jump Markov chain Monte Carlo (MCMC) (28), beam search with a priority queue limited by a constant bound, and a search that is not guided by heuristics (comparable to Dijkstra's algorithm for planning). For consistency, we use the softmin distance to generate the values of each state in a sampled MCMC trajectory. If $\mathrm{d}_{\text{soft}}(\mathrm{s}) = \infty$, we use the optimistic heuristic defined in Equation 5.1 generating a lower bound on d_{soft}.

5.2 Character drawing

We apply our approach to the task of predicting the sequential pen strokes employed to draw different characters from the Latin alphabet. Despite the apparent simplicity of these tasks, applying standard inverse optimal control methods is challenging due to the large planning graph that corresponds to a fine-grained representation of the task. We demonstrate the overall effectiveness of both the *SoftStar* algorithm and heuristic-guided value iteration (heuristic sampling) in learning a model that provides accurate predictions against other commonly employed techniques.

5.2.1 Demonstrated data

A handwritten character, such as the one depicted in Figure 1, is converted into a skeleton of nodes within a unit character frame. The character in Figure 1 was drawn using two strokes, red and green respectively. The numbering indicates the start of each stroke (42).

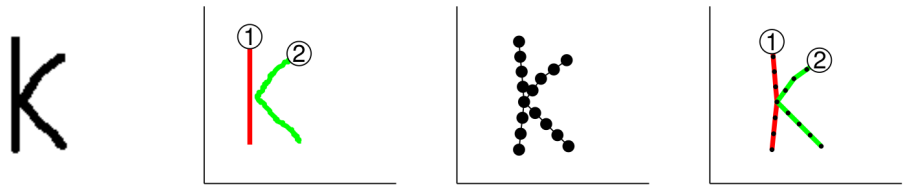

Figure 1: Demonstrated construction of the character ‘K’ showing the sequence of pen strokes and the nodal representation.

The data consists of a randomly separated training set of 400 drawn characters, each with a unique demonstrated trajectory, and a separate test set of 52 examples.

5.2.2 State and feature representation

The state consists of a two node history (previous node and current node) and a bitmap signifying which edges are covered/uncovered. The size of the state space is $2^{|\mathrm{E}|}(|V|+1)^2$ with $|\mathrm{E}|$ edges and $|V|$ nodes. The number of nodes is increased by one to account for the initial state. For example, the character in Figure 1 has 16 nodes and 15 edges. It has a corresponding state space of about 9.47 million states.

The initial state has no previously traversed nodes and a bitmap with all uncovered edges. The goal state will have a two node history as defined above, and a fully set bitmap representing all of the required edges being covered. Any transition between nodes in the skeleton is allowed, with transitions between neighboring nodes characterized as edge draws and all other transitions as pen lifts.

There are 28 Euclidean features including inverse features that improve the fine tuning of the preference relationships. The features are separated into three groups:

- **Four initial transition features:** Destination node distance from each edge.
- **Nine edge draw features:** Angles of the edge being drawn, with respect to the previous transition angle, to the horizontal, to the vertical, the inverse of these 4 features, and a constant edge draw feature.
- **15 pen lift features:** Each of the edge draw features and initial transition features as well as the distance of the lift and the inverse of the distance of the lift.

5.2.3 Heuristic

We consider a heuristic function that combines the (soft) minimum costs of covering each remaining uncovered edge in a character assuming all moves that do not cross an uncovered edge have zero cost. Formally, it is expressed as

$$\hat{\mathrm{h}}_{\mathsf{soft}}(\mathrm{s}) = \sum_{\mathrm{e_i} \in \mathrm{E_u}} \underset{\mathrm{e_i}}{\mathrm{softmin}}\, \mathrm{cost}(\mathrm{e_i}), \tag{5.1}$$

with $\mathrm{E_u}$ representing the set of uncovered edges and $\mathrm{cost}(\mathrm{e_i})$ the set of all possible costs of traversing edge i.

5.2.4 Estimated parameters

The learned weights indicate that the preferred starting position for a character is in the top left of the frame, drawing an edge is more desirable than a pen lift and that a smoother transition angle is preferred when drawing an edge than when executing a pen lift matching previous empirical results on drawing preferences (27).

5.3 Professional Soccer

In addition, we apply our approach to the task of modeling the discrete spatial decision process of the ball-handler for single possession open plays in professional soccer. As in the case for the character drawing task, we demonstrate the overall effectiveness of our against other commonly employed techniques.

5.3.1 Demonstrated data

The data contains tracking information from 93 games consisting of player locations and time steps of significant events/actions. We pre-processed the data into sets of sequential actions in single possessions. Each possession may include multiple different team-mates handling the ball at different times resulting in a team decision process on the ball actions rather than single player actions/decisions.

Due to the nature of the task, it is necessary to discretize the soccer field into cells leading to a very large decision process when considering actions to each cell at each step. To increase generalization and the relevance of the training data, we reformatted the field coordinates so that the origin lies in the center of the possessing teams goal and all playing fields are normalized to a standard 105m by 68m, discretized into 5x4m cells. Formatting the field coordinates based on the target, x/y distances from the goal, of the team in possession doubles the amount of training data for similar coordinates. The positive and negative half planes of the y axis capture which side of the goal the ball is located on.

We train a spatial decision model on 92 of the games and evaluate the learned ball trajectories on a single test game. Once formatted and pre-processed the data contains 20,337 training possession sequences and 230 test sequences.

5.3.2 State and feature representation

The state consists of a two action history where an action is designated as a type-cell tuple where the type is the action type (pass, shot, clear, dribble, or cross) and the cell is the destination cell with the most recent action containing the current cell location of the ball.

Limiting shot actions to only be aimed at the goal gives us an action space of 1433 possible actions at each step in a trajectory resulting in about 2.05 million possible states.

There are 28 Euclidean features for each action type and 29 that apply to all action types resulting in 168 total features. We use the same features as the character drawing model and include a different set of features for each action type in order to learn unique action based cost functions.

5.3.3 Heuristic

We use the softmin cost over all possible actions from the current state as a heuristic. It is guaranteed to be admissible as it assumes that the next state will always be the goal: $\hat{\mathrm{h}}_{\mathrm{soft}}(\mathrm{s}) = \underset{\mathrm{s}' \in \mathcal{N}(\mathrm{s})}{\mathrm{softmin}} \left\{ \mathrm{cost}(\mathrm{s}, \mathrm{s}') \right\}$.

5.3.4 Estimated parameters

The learned weights indicate that the players prefer close range shots in front of the goal, crosses from the sides of the field, and players tend to take actions that move the ball closer to the goal in short distances. A more detailed analysis of learned behavior will be reserved for future work.

5.4 Results

5.4.1 Learning efficiency

We compare *SoftStar* and heuristic sampling to another inference procedure estimating path distributions in large scale IOC and measure the average test set log-loss, equivalent to the difference between the cost of the demonstrated path, $\mathrm{cost}(\mathrm{s}_{1:\mathrm{T}})$, and the softmin distance to the goal, $\mathrm{d}_{\mathrm{soft}}(\mathrm{goal})$, $-\log \mathrm{P}(\mathrm{path}) = \mathrm{cost}(\mathrm{s}_{1:\mathrm{T}}) - \mathrm{d}_{\mathrm{soft}}(\mathrm{goal})$.

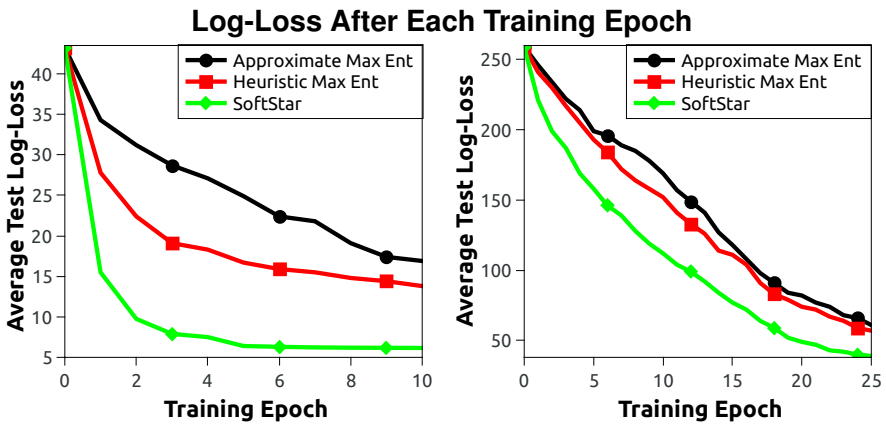

Figure 2: Training efficiency experiments on the Character domain (left) and the Soccer domain (right).

Figure 2 shows the decrease of the test set log-loss after each training epoch. *SoftStar* learns the models far more efficiently than both approximate max ent IOC (32) and heuristic guided sampling (Algorithm 3). This is likely due to the more accurate estimation of the feature expectations that results from searching the graph rather than sampling trajectories.

The improved efficiency of the *SoftStar* method is also evident if we analyze the respective time taken to train each model. *SoftStar* took ~5 hours to train 10 epochs for the character model and ~12 hours to train 25 epochs for the soccer model. To compare, heuristic sampling took ~9 hours for the character model and ~17 hours for the soccer model, and approximate max ent took ~10 hours for the character model and ~20 hours for the soccer model.

5.4.2 Inference efficiency

In addition to evaluating our learning efficiency, we compare the time efficiency for generating lower bounds on the estimated softmin distance to the goal on a single random test example from each model in Figure 3.

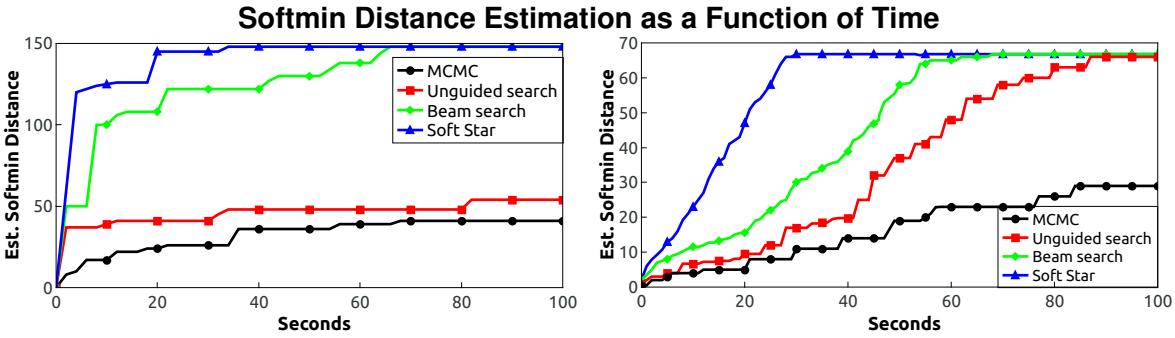

Figure 3: Inference efficiency evaluations for the Character domain (left) the Soccer domain (right).

The results for both models are comparable to those found using the synthetic data in Figure 2. The MCMC approach has trouble with local optima where the estimated softmin distance has difficulty improving at a competitive rate. While the unguided algorithm ($\alpha = 0\%$) does not experience this problem, it instead explores a large number of improbable paths to the goal. This exhaustive search is both time consuming and memory intensive. Beam

search performs better than the unguided algorithm, but suffers from a lack of completeness guarantees where it may never find a solution (65).

The *SoftStar* algorithm avoids low probability paths and converges efficiently converging much faster than the comparison methods. MCMC fails to converge on both examples even after 1,200 seconds, matching past experience with the character data[1]. It should be noted that while these examples were chosen at random, average convergence time on the test set characters after training was about 12 seconds and 21 seconds for the soccer data using SoftStar.

5.5 Discussion

We introduced a heuristic-guided inference algorithm that used sampling-based softened value iteration to efficiently infer near-optimal path distributions through state-space graphs. We then extended this idea to create a heuristic-guided search technique for estimating path distributions with approximation guarantees in both deterministic and stochastic decision graphs.

We validated our proposed methods on two complex decision problems for inverse optimal control (IOC) and found significant performance improvements compared to other IOC inference methods. This is especially evident in the *SoftStar* algorithm where the increase in information generated from the approximation guarantees results in a much more informed gradient update and significantly faster convergence in optimizing the cost parameters.

---

[1] Previous inference methods (e.g., MCMC) were incapable of the efficiency needed to enable learning without first employing significant simplifying abstractions to the state representation

Probabilistic search in these settings is significantly more computationally demanding than A* search, both in theory and practice, primarily due to key differences between the $\min$ and $\mathrm{softmin}$ functions. However, this increase in complexity is necessary to approximate the distribution of near optimal paths to the goal rather than simply finding the shortest path.

A weakness of the heuristic search algorithms presented lies in the need to discretize and search the state space which can be prohibitive in problems that require real-time solutions. With this in mind, the following chapters will present two continuous IOC methods for real time prediction and planning in complex scenes.

Part II

Inverse Linear Quadratic Regulation

Partially published in the Association for the Advancement of Artificial Intelligence Conference (http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9897) (52).

Depth cameras, like the Microsoft Kinect, have been used to detect human activities (38; 58; 67) and provide rich information including human skeleton movement data and 3D environment maps. Critical to this task are the roles of high-dimensionality and uncertainty. Many co-robotics tasks are performed within high-dimensional control spaces where there are a variety of ways for a human to reasonably accomplish the task. Behavior modeling techniques for intent recognition and trajectory forecasting must scale to these high-dimensional control spaces and incorporate uncertainty over inherently ambiguous human behavior.

The problem with discrete inference procedures is that the discretization of a high-dimensional state space results in a very inefficient IOC method that becomes infeasible in conditions where real time solutions are required. For instance, there has been an increasing desire for co-robotic applications that situate robots as partners with humans in cooperative and tightly interactive tasks (73; 66; 34). Unlike previous generations of human-robot interaction applications, which might need to respond to recognized human behavior (60), co-robotics requires robots to act in anticipation of future human behavior to realize the desired levels of seamless interaction.

Additionally, the desirable space of manipulation trajectories in typical robotic arms with many degrees of freedom (DOF) is often multi-modal and non-linear due to collision avoidance with discrete obstacles. However, discrete planners fail to capture continuous motion dynamics and require discretizing high-dimensional joint angles.

In this chapter we present an inverse linear quadratic regulation algorithms for efficiently modeling continuous trajectories for task prediction, forecasting and planning. The first section covers an inverse linear quadratic regulation method for predicting household activities and target objects while inferring a distribution of predicted future trajectories. In the next section we extend the inverse linear quadratic regulation model from the first section to bind the continuous distribution around discrete feature representations, in the form of inferred discrete paths learned from demonstrated behavior. This allows for a model to incorporate discrete features, such as obstacles, while avoiding the inherent inefficiency of planning full discrete trajectories.

CHAPTER 6

INTENT PREDICTION VIA INVERSE LINEAR QUADRATIC REGULATION

Previously published in the Association for the Advancement of Artificial Intelligence Conference (http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9897) (52).

In this chapter we present a continuous inverse optimal control (IOC) approach using Linear-Quadratic Regulation (LQR) for intention recognition and trajectory forecasting of tasks involving hand motion trajectories. We utilize the recently developed technique of maximum entropy IOC with LQR (78; 77; 46), allowing us to efficiently apply the inverse optimal control approach to continuous-valued three-dimensional positions, velocities, accelerations, etc.

We base our approach on prior work on inverse linear quadratic regulation for predicting computer cursor targets (77). We extend this method by incorporating an additional learned penalty to the final state for deviating far from the desired goal state features and apply it to the 3-dimensional problem of predicting human activities from depth videos. This formulation is inherently probabilistic and enables the inference of the user's intent based on partial behavior observations and forecasts of future behavior.

We evaluate our results on the Cornell Activity Dataset (CAD-120) (37) and compare to the previous state-of-the-art approach for detecting/predicting human sub-activities.

6.1 Related Work

There has been a significant amount of recent work on forecasting the future behavior of people to improve intelligent systems. In the robotic navigation domain, this has been manifested in robots that plan paths that are complementary to a pedestrian's future movements (83) or navigate through crowds based on anticipated movements (31; 74; 39). In robotic manipulation, techniques that interpret and aid in realizing a teleoperator's intentions to complete a task (30) have had success.

Our work is most closely related, but complementary, to anticipatory temporal conditional random fields (ATCRF) (37). Under that approach, discriminative learning is employed to model the relationships between object affordances and sub-activities at the "discrete" level and a simple generative model (based on a Gaussian distribution) of human pose and object location trajectories is employed at the "continuous" level.

We extend discriminative learning techniques, in the form of inverse optimal control, to the continuous level of human pose trajectories. The two approaches are complementary in that any inferred object affordances and sub-activities at the discrete level can be employed to shape the prior distributions at the continuous level, and the posterior inferences at the continuous-level can feed into inferences at the discrete level. Our approach differs in its applicability to continuous-valued control settings and in employing a maximum entropy formulation rather than a non-probabilistic maximum-margin estimation approach.

Maximum entropy inverse optimal control (81) has been shown to elicit strong results by maximizing the uncertainty in the probabilistic inference task (82). Extensions to the linear-

quadratic setting have been applied to predicting the intended targets of computer cursor movements (77) detailing the benefit of using demonstrated continuous trajectories for target prediction. There has also been recent work in the area of action inference and activity forecasting from video (36), however this work discretizes the state space in such a way that would be impractical for inferring at useful granularities in higher dimensional (three or more) spaces.

We extend the maximum entropy inverse optimal control linear quadratic regulator model (77) to the task of predicting target intentions and inferring continuous hand motion trajectories using depth camera data. This is directly applicable to the areas of activity prediction (37) and behavior forecasting (83).

6.2 Approach

We propose a predictive model of motion trajectories trained from a dataset of observed depth camera motions using a linear quadratic regulation framework. Under this framework, a linear state dynamics model and a quadratic cost function are assumed.

We consider the state of the hand, $\mathrm{s_t}$, to be represented by its position, velocity, and acceleration, and control actions to be represented by the velocity vector, which is the change of position from current state to the next state. The state transition dynamics follow a linear relationship in this setting and we additionally assume the cost function to be quadratic with respect to state and action. The optimal control problem for this formulation is to find the best control action for each possible state that minimizes the expectation of the cost function over time. In contrast, we estimate the cost function that best rationalizes demonstrated trajectories

in this work. Our approach provides a probabilistic model that allows us to reason about the intentions of partially completed trajectories and generate an optimal trajectory from the origin to the target.

6.2.1 State Representation and Quadratic Cost Matrices

One of the main benefits of the proposed technique is the ability to use a simple and generalizable state model. For the task at hand, we consider the xyz positions of the center of the hand as well as the velocities and accelerations in each respective axis. Adding one constant term to capture the temporal, linear and state invariant costs gives us a 10-dimensional state representation at time t,

$$\mathbf{s}_\mathrm{t} = [\mathrm{x}_\mathrm{t}, \mathrm{y}_\mathrm{t}, \mathrm{z}_\mathrm{t}, \dot{\mathrm{x}}_\mathrm{t}, \dot{\mathrm{y}}_\mathrm{t}, \dot{\mathrm{z}}_\mathrm{t}, \ddot{\mathrm{x}}_\mathrm{t}, \ddot{\mathrm{y}}_\mathrm{t}, \ddot{\mathrm{z}}_\mathrm{t}, 1]^\mathrm{T},$$

where $(\dot{\mathrm{x}}_\mathrm{t}, \dot{\mathrm{y}}_\mathrm{t}, \dot{\mathrm{z}}_\mathrm{t})$ represent the velocities, and $(\ddot{\mathrm{x}}_\mathrm{t}, \ddot{\mathrm{y}}_\mathrm{t}, \ddot{\mathrm{z}}_\mathrm{t})$ represent the accelerations. A constant of 1 is added to the state representation in order to incorporate linear features into the quadratic cost function formulation described later.

Importantly, when actions, $\mathbf{a}_\mathrm{t}$ represent the velocities specifying changes in positions between current state and next state, the state dynamics follow a linear relationship: $\mathbf{s}_{\mathrm{t}+1} = \mathbf{A}\mathbf{s}_\mathrm{t} + \mathbf{B}\mathbf{a}_\mathrm{t} + \epsilon_\mathrm{t}$, where the noise, ϵ_t, is drawn from a zero-mean Gaussian distribution. We represent the distribution over next state using the transition probability distribution $\mathrm{P}(\mathbf{s}_{\mathrm{t}+1}|\mathbf{s}_\mathrm{t}, \mathbf{a}_\mathrm{t})$. Though many tasks may have non-linear dynamics or be motivated by non-quadratic cost functions, the LQR simplification is important when considering the fast execution time needed for true human-robot interaction. If needed, additional features can be added to the state in order to further incorporate higher-order attributes of the trajectory, like jerks. In addition

to efficiency, this state definition provides a strong set of informative features for defining (or estimating) a cost function. While the position is clearly correlated with the intended target, the motion dynamics offer a more distinct descriptive behavior in terms of completing differing tasks (25).

In order to account for the high level of variance with respect to the xyz goal positions of the demonstrated trajectories obtained from the depth camera data, we extend the LQR method (77) and add an additional parameter matrix to the existing cost matrix $\mathbf{M}$. We call this matrix $\mathbf{M_f}$ to signify that it only applies to the final state of the trajectory. This set of parameters adds an additional penalty to the final state for deviating far from the desired goal state features. So the cost functions are:

$$\mathrm{cost}(\mathbf{s}_t, \mathbf{a}_t) = \begin{bmatrix} \mathbf{a}_t \\ \\ \mathbf{s}_t \end{bmatrix}^T \mathbf{M} \begin{bmatrix} \mathbf{a}_t \\ \\ \mathbf{s}_t \end{bmatrix}, t < T, \tag{6.1}$$

$$\mathrm{cost}(\mathbf{s}_T) = (\mathbf{s}_T - \mathbf{s}_G)^T \mathbf{M_f} (\mathbf{s}_T - \mathbf{s}_G). \tag{6.2}$$

To increase the generalization of the model to all possible xyz conditions, we sparsify the two parameter matrices so that we only train parameters relating to the dynamics features, velocity and acceleration, for $\mathbf{M}$, and only train parameters relating to xyz positions for $\mathbf{M_f}$. Therefore, all quadratic feature combinations that include an xyz position feature in $\mathbf{M}$ or a velocity, or acceleration, feature in $\mathbf{M_f}$ will have a parameter value of 0. This is helpful in the case where the xyz coordinates differ greatly across the demonstrated trajectories and allows

for $\mathbf{M}$ to learn the behavior of the trajectories rather than the true positions. Likewise, $\mathbf{M_f}$ can then serve solely as a position penalty for the final inferred position deviating from the desired goal location and avoid penalizing the velocity and acceleration of the final state. This helps to generalize the model across different orientations and behaviors.

6.2.2 Inverse Linear-Quadratic Regulation for Prediction

We employ maximum causal entropy inverse optimal control (79) in the linear quadratic regulation setting to obtain estimates for our cost parameter matrices, $\mathbf{M}$ and $\mathbf{M}_\mathrm{f}$. These estimates result from a constrained optimization problem maximizing causal entropy,

$$\mathrm{H}(\mathbf{a}\|\mathbf{s}) \triangleq \mathbb{E}_{\hat{\pi}}\left[-\sum_{t=1}^{T}\log\hat{\pi}(\mathbf{a}\|\mathbf{s})\right],$$

so that the predictive policy distribution, $\hat{\pi}(\mathbf{a}\|\mathbf{s}) = \pi(\mathbf{a}_1|\mathbf{s}_1)\pi(\mathbf{a}_2|\mathbf{s}_2)\cdots\pi(\mathbf{a}_T|\mathbf{s}_T)$, matches the quadratic state properties of the demonstrated behavior, $\tilde{\pi}$, in feature expectation where:

$$\mathbb{E}_{\hat{\pi}}\left[\sum_{t=1}^{T-1}\begin{bmatrix}\mathbf{a}_t\\ \mathbf{s}_t\end{bmatrix}\begin{bmatrix}\mathbf{a}_t\\ \mathbf{s}_t\end{bmatrix}^{T}\right] = \mathbb{E}_{\tilde{\pi}}\left[\sum_{t=1}^{T-1}\begin{bmatrix}\mathbf{a}_t\\ \mathbf{s}_t\end{bmatrix}\begin{bmatrix}\mathbf{a}_t\\ \mathbf{s}_t\end{bmatrix}^{T}\right] \text{ and} \tag{6.3}$$

$$\mathbb{E}_{\hat{\pi}}\Big[(\mathbf{s}_T-\mathbf{s}_G)(\mathbf{s}_T-\mathbf{s}_G)^{T}\Big] = \mathbb{E}_{\tilde{\pi}}\Big[(\mathbf{s}_T-\mathbf{s}_G)(\mathbf{s}_T-\mathbf{s}_G)^{T}\Big].$$

This set of constraints ensures that the trajectory's dynamic properties are maintained by the control policy estimate, $\hat{\pi}$. Solving this problem, we obtain a state-conditioned probabilistic policy $\hat{\pi}$ over actions that are recursively defined using the following equations:

$$\hat{\pi}(\mathbf{a}_t|\mathbf{s}_t) = e^{Q(\mathbf{s}_t,\mathbf{a}_t)-V(\mathbf{s}_t)}, \tag{6.4}$$

$$Q(\mathbf{s}_t,\mathbf{a}_t) = \mathbb{E}_{\tau(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}[V(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)] + cost(\mathbf{s}_t,\mathbf{a}_t), \tag{6.5}$$

$$V(\mathbf{s}_t) = \begin{cases} \underset{\mathbf{a}_t}{\operatorname{softmax}}\, Q(\mathbf{s}_t,\mathbf{a}_t), & t < T \\ (\mathbf{s}_t - \mathbf{s}_G)^T \mathbf{M}_\mathbf{f}(\mathbf{s}_t - \mathbf{s}_G), & t = T, \end{cases} \tag{6.6}$$

where $\mathbf{s}_G$ represents a goal state that is used to specify a penalty for a trajectory's final distance from the desired goal and the $\operatorname{softmax}$ function is a smoothed interpolation of the maximum function, $\underset{x}{\operatorname{softmax}}\, f(x) = \log \int_x e^{f(x)} dx$.

Using this formulation, the recurrence of Equation 6.5 and Equation 6.6 can be viewed as a probabilistic relaxation of the Bellman criteria for optimal control; it selects actions with smaller expected future costs in Equation 6.5 and recursively computes those future costs using the expectation over the decision process's dynamics in Equation 6.6. The Q and V functions are softened versions of the optimal control state-action and state value functions, respectively. Inference is performed by forming a set of time-specific update rules that are used to compute state and action values which are conditioned on the goal state, trajectory length, and the cost parameter values. The probability of a trajectory of length T is then easily computed as:

$\prod_{\tau=1}^{t} \pi(a_t|s_t, G, I, T) = e^{\sum_{\tau=1}^{t-1} Q(a_t,s_t)-V(s_t)}$, where G is the goal location of the trajectory and I is the characteristic intention/behavior of the movement (ex. reaching, placing, eating, etc.).

The recursive Q and V calculations simplify to a set of matrix updates in the LQR setting:

$$Q(\mathbf{s}_t, \mathbf{a}_t) = \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \begin{bmatrix} \mathbf{C}_{a_t,a_t} & \mathbf{C}_{a_t,s_t} \\ \mathbf{C}_{s_t,a_t} & \mathbf{C}_{s_t,s_t} \end{bmatrix} \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix} + \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \begin{bmatrix} \mathbf{F}_{a_t} \\ \mathbf{F}_{s_t} \end{bmatrix} + Q_{cv_t},$$

$$V(\mathbf{s}_t) = \mathbf{s}_t^T \mathbf{D}_t \mathbf{s}_t + \mathbf{s}_t^T \mathbf{G}_t + V_{cv_t},$$

where Q_{cv_t} and V_{cv_t} are scalars. The matrices of these values are recursively defined as:

$$\mathbf{D}_T = \mathbf{M}_f; \qquad \mathbf{G}_T = -2\mathbf{M}_f \mathbf{s}_G;$$

for t in $T-1$...1

$$\mathbf{C}_{a_t,a_t} = \mathbf{B}^T \mathbf{D}_{t+1} \mathbf{B} + \mathbf{M}_{a,a}; \mathbf{C}_{a_t,s_t} = \mathbf{B}^T \mathbf{D}_{t+1} \mathbf{A} + \mathbf{M}_{a,s};$$

$$\mathbf{C}_{s_t,a_t} = \mathbf{A}^T \mathbf{D}_{t+1} \mathbf{B} + \mathbf{M}_{s,a}; \mathbf{C}_{s_t,s_t} = \mathbf{A}^T \mathbf{D}_{t+1} \mathbf{A} + \mathbf{M}_{s,s};$$

$$\mathbf{F}_{a_t} = \mathbf{B}^T \mathbf{G}_{t+1}; \qquad \mathbf{F}_{s_t} = \mathbf{A}^T \mathbf{G}_{t+1};$$

$$\mathbf{D}_t = \mathbf{C}_{s_{t+1},s_{t+1}} - \mathbf{C}^T_{a_{t+1},s_{t+1}} \mathbf{C}^{-1}_{a_{t+1},a_{t+1}} \mathbf{C}_{a_{t+1},s_{t+1}};$$

$$\mathbf{G}_t = \mathbf{F}_{s_{t+1}} - \mathbf{C}^T_{a_{t+1},s_{t+1}} \mathbf{C}^{-1}_{a_{t+1},a_{t+1}} \mathbf{F}_{a_{t+1}},$$

which are derived in the following section.

6.2.2.1 Update Rule Derivation

Since recurrence values in Equations Equation 6.5 and Equation 6.6 are of quadratic forms, we break down the matrices into time dependent components:

$$Q(\mathbf{s}_t, \mathbf{a}_t) = \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \begin{bmatrix} \mathbf{C}_{a_t,a_t} & \mathbf{C}_{a_t,s_t} \\ \mathbf{C}_{s_t,a_t} & \mathbf{C}_{s_t,s_t} \end{bmatrix} \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix} + \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \begin{bmatrix} \mathbf{F}_{a_t} \\ \mathbf{F}_{s_t} \end{bmatrix} + Q_{cv_t},$$

$$V(\mathbf{s}_t) = \mathbf{s}_t^T \mathbf{D}_t \mathbf{s}_t + \mathbf{s}_t^T \mathbf{G}_t + V_{cv_t},$$

where Q_{cv_t} and V_{cv_t} are the respective constants derived from the Q and V functions at time t.

Combining the above definitions with Equations Equation 6.5 and Equation 6.6 yields:

$$
\begin{aligned}
Q(\mathbf{s}_t, \mathbf{a}_t) &= \mathbb{E}_{\tau(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}[\mathbf{s}_{t+1}^T \mathbf{D}_t \mathbf{s}_{t+1} + \mathbf{s}_{t+1}^T \mathbf{G}_t + V_{cv_t}|\mathbf{s}_t, \mathbf{a}_t] \\
&+ cost(\mathbf{s}_t, \mathbf{a}_t), \\
&= (\mathbf{A}\mathbf{s}_t + \mathbf{B}\mathbf{a}_t)^T \mathbf{D}_t (\mathbf{A}\mathbf{s}_t + \mathbf{B}\mathbf{a}_t) + tr(\mathbf{D}_t \Sigma) \\
&+ \mathbf{a}_t^T \mathbf{B}^T \mathbf{G}_t + \mathbf{s}_t^T \mathbf{A}^T \mathbf{G}_t + \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \mathbf{M} \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \begin{bmatrix} \mathbf{B}^T \mathbf{D}_t \mathbf{B} & \mathbf{B}^T \mathbf{D}_t \mathbf{A} \\ \mathbf{A}^T \mathbf{D}_t \mathbf{B} & \mathbf{A}^T \mathbf{D}_t \mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix} \\
&+ \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \begin{bmatrix} \mathbf{M}_{a_t,a_t} & \mathbf{M}_{a_t,s_t} \\ \mathbf{M}_{s_t,a_t} & \mathbf{M}_{s_t,s_t} \end{bmatrix} \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix} + \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \begin{bmatrix} \mathbf{B}^T \mathbf{G}_t \\ \mathbf{A}^T \mathbf{G}_t \end{bmatrix}.
\end{aligned}
$$

When $t < T$,

$$
\begin{aligned}
&V(\mathbf{s}_t) = \\
&\ln \int_{\mathbf{a}_t} e^{\mathbf{a}_t^T \mathbf{C}_{a_t,a_t} \mathbf{a}_t + 2\mathbf{a}_t^T \mathbf{C}_{a_t,s_t} \mathbf{s}_t + \mathbf{s}_t^T \mathbf{C}_{s_t,s_t} \mathbf{s}_t + \mathbf{s}_t^T \mathbf{F}_{s_t} + \mathbf{a}_t^T \mathbf{F}_{a_t} + Q_{cv_t}} d\mathbf{a}_t \\
&= \mathbf{s}_t^T \mathbf{C}_{s_t,s_t} \mathbf{s}_t + \mathbf{s}_t^T \mathbf{F}_{s_t} + Q_{cv_t} \\
&+ \ln(\frac{1}{Z_t}) \int_{\mathbf{a}_t} Z_t e^{-0.5 \mathbf{a}_t^T (-2\mathbf{C}_{a_t,a_t}) \mathbf{a}_t + \mathbf{a}_t^T (2\mathbf{C}_{a_t,s_t} \mathbf{s}_t + \mathbf{F}_{a_t})} d\mathbf{a}_t \\
&= \mathbf{s}_t^T \mathbf{C}_{s_t,s_t} \mathbf{s}_t + \mathbf{s}_t^T \mathbf{F}_{s_t} + Q_{cv_t} \\
&+ \ln(\frac{1}{Z_t}) \int_{\mathbf{a}_t} N[\mathbf{a}_t | 2\mathbf{C}_{a_t,s_t} \mathbf{s}_t + \mathbf{F}_{a_t}, -2\mathbf{C}_{a_t,a_t}] d\mathbf{a}_t \\
&= \mathbf{s}_t^T \mathbf{C}_{s_t,s_t} \mathbf{s}_t + \mathbf{s}_t^T \mathbf{F}_{s_t} + Q_{cv_t} - \ln Z_t,
\end{aligned}
$$

where

$$
Z_t = \frac{e^{(-\frac{1}{2})(2\mathbf{C}_{a_t,s_t} \mathbf{s}_t + \mathbf{F}_{a_t})^T (-2\mathbf{C}_{a_t,a_t})^{-1} (2\mathbf{C}_{a_t,s_t} \mathbf{s}_t + \mathbf{F}_{a_t})}}{|2\pi(-2\mathbf{C}_{a_t,a_t})^{-1}|^{1/2}}.
$$

$$
\begin{aligned}
V(\mathbf{s}_t) = & \mathbf{s}_t^T(\mathbf{C}_{s_t,s_t} - \mathbf{C}_{a_t,s_t}^T \mathbf{C}_{a_t,a_t}^{-1} \mathbf{C}_{a_t,s_t})\mathbf{s}_t \\
& + \mathbf{s}_t^T(\mathbf{F}_{s_t} - \mathbf{C}_{a_t,s_t}^T \mathbf{C}_{a_t,a_t}^{-1} \mathbf{F}_{a_t}) \\
& + Q_{cv_t} + \frac{1}{2}\ln|2\pi(-2C_{a_t,a_t})^{-1}| - \frac{1}{4}\mathbf{F}_{a_t}^T \mathbf{C}_{a_t,a_t}^{-1} \mathbf{F}_{a_t} \\
= & \mathbf{s}_t^T(\mathbf{C}_{s_t,s_t} - \mathbf{C}_{a_t,s_t}^T \mathbf{C}_{a_t,a_t}^{-1} \mathbf{C}_{a_t,s_t})\mathbf{s}_t \\
& + \mathbf{s}_t^T(\mathbf{F}_{s_t} - \mathbf{C}_{a_t,s_t}^T \mathbf{C}_{a_t,a_t}^{-1} \mathbf{F}_{a_t}) + Q_{cv_t} \\
& + \frac{1}{2}\ln|2\pi(-2C_{a_t,a_t})^{-1}| - \frac{1}{4}\mathbf{F}_{a_t}^T \mathbf{C}_{a_t,a_t}^{-1} \mathbf{F}_{a_t},
\end{aligned}
$$

allowing for,

$$
\begin{aligned}
\mathbf{D}_t &= \mathbf{C}_{s_t,s_t} - \mathbf{C}_{a_t,s_t}^T \mathbf{C}_{a_t,a_t}^{-1} \mathbf{C}_{a_t,s_t}, \\
\mathbf{G}_t &= \mathbf{F}_{s_t} - \mathbf{C}_{a_t,s_t}^T \mathbf{C}_{a_t,a_t}^{-1} \mathbf{F}_{a_t},
\end{aligned}
$$

with,

$$
V_{cv_t} = Q_{cv_t} + \frac{1}{2}\ln|2\pi(-2C_{a_t,a_t})^{-1}| - \frac{1}{4}\mathbf{F}_{a_t}^T \mathbf{C}_{a_t,a_t}^{-1} \mathbf{F}_{a_t}.
$$

The set of update rules for the quadratic functions are:

$$\mathbf{D}_T = \mathbf{M}_f;$$

$$\mathbf{G}_T = -2\mathbf{M_f s}_G;$$

for t in $T-1...1$

$$\mathbf{C}_{a_t,a_t} = \mathbf{B}^T\mathbf{D}_{t+1}\mathbf{B}+\mathbf{M}_{a_{t+1},a_{t+1}};$$

$$\mathbf{C}_{a_t,s_t} = \mathbf{B}^T\mathbf{D}_{t+1}\mathbf{A}+\mathbf{M}_{a_{t+1},s_{t+1}};$$

$$\mathbf{C}_{s_t,a_t} = \mathbf{A}^T\mathbf{D}_{t+1}\mathbf{B}+\mathbf{M}_{s_{t+1},a_{t+1}};$$

$$\mathbf{C}_{s_t,s_t} = \mathbf{A}^T\mathbf{D}_{t+1}\mathbf{A}+\mathbf{M}_{s_{t+1},s_{t+1}};$$

$$\mathbf{F}_{a_t} = \mathbf{B}^T\mathbf{G}_{t+1};$$

$$\mathbf{F}_{s_t} = \mathbf{A}^T\mathbf{G}_{t+1};$$

$$\mathbf{D}_t = \mathbf{C}_{s_{t+1},s_{t+1}} - \mathbf{C}^T_{a_{t+1},s_{t+1}}\mathbf{C}^{-1}_{a_{t+1},a_{t+1}}\mathbf{C}_{a_{t+1},s_{t+1}};$$

$$\mathbf{G}_t = \mathbf{F}_{s_{t+1}} - \mathbf{C}^T_{a_{t+1},s_{t+1}}\mathbf{C}^{-1}_{a_{t+1},a_{t+1}}\mathbf{F}_{a_{t+1}}.$$

The model parameters in matrix M and $\mathrm{M_f}$ are fit from the demonstrated examples. The likelihood of a demonstration is a convex function of M and $\mathrm{M_f}$ guaranteeing gradient optimization converges to paramters that best explain the demonstrations (77). Under the maximum

entropy framework, the gradients have intuitive interpretation: they are just the differences of the optimization constraints,

$$\nabla_M L = \mathbb{E}_{\hat{\pi}} \left[\sum_{t=1}^{T} \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix} \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \right] - \mathbb{E}_{\tilde{\pi}} \left[\sum_{t=1}^{T} \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix} \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \right],$$

$$\nabla_{M_f} L = \mathbb{E}_{\hat{\pi}} \left[(\mathbf{s}_T - \mathbf{s}_G)(\mathbf{s}_T - \mathbf{s}_G)^T \right] - \mathbb{E}_{\tilde{\pi}} \left[(\mathbf{s}_T - \mathbf{s}_G)(\mathbf{s}_T - \mathbf{s}_G)^T \right].$$

which can be calculated via the state Gaussian distribution (77). Concretely, if x is normally distributed with mean μ_x and covariance Σ_x, $\mathbb{E}[xx^T] = \mu_x \mu_x^T + \Sigma_x$. Here, according to the Gaussian properties, if the distribution of s_t is $N(\mu_{s_t}, \Sigma_{s_t})$:

$$\begin{aligned}
&\log(\hat{\pi}(\mathbf{a}_t|\mathbf{s}_t)) = Q(\mathbf{s}_t, \mathbf{a}_t) - V(\mathbf{s}_t) \\
&= \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \begin{bmatrix} \mathbf{C}_{a_t,a_t} & \mathbf{C}_{a_t,s_t} \\ \mathbf{C}_{s_t,a_t} & \mathbf{C}_{s_t,s_t} \end{bmatrix} \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix} + \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \begin{bmatrix} \mathbf{F}_{a_t} \\ \mathbf{F}_{s_t} \end{bmatrix} - \mathbf{s}_t^T \mathbf{D}_t \mathbf{s}_t - \mathbf{s}^T \mathbf{G}_t \\
&- \frac{1}{2} \log |2\pi(-2C_{a_t,a_t})^{-1}| + \frac{1}{4} \mathbf{F}_{a_t}^T \mathbf{C}_{a_t,a_t}^{-1} \mathbf{F}_{a_t},
\end{aligned}$$

which means $\hat{\pi}(\mathbf{a}_t|\mathbf{s}_t)) \propto N[\mathbf{a}_t|2\mathbf{C}_{a_t,s_t}\mathbf{s}_t + \mathbf{F}_{a_t}, -2\mathbf{C}_{a_t,a_t}]$,

which is $N((-\mathbf{C}_{a_t,a_t})^{-1}\mathbf{C}_{a_t,s_t}\mathbf{s}_t - \frac{1}{2}\mathbf{C}_{a_t,a_t}^{-1}\mathbf{F}_{a_t}, (-2\mathbf{C}_{a_t,a_t})^{-1})$.

So $a_t \sim N(\mu_{a_t}, \Sigma_{a_t})$ where:

$$\mu_{a_t} = -\mathbf{C}_{a_t,a_t}^{-1}\mathbf{C}_{a_t,s_t}\mu_{s_t} - \frac{1}{2}\mathbf{C}_{a_t,a_t}^{-1}\mathbf{F}_{a_t},$$

$$\Sigma_{a_t} = (-2\mathbf{C}_{a_t,a_t})^{-1}$$

$$+(-\mathbf{C}_{a_t,a_t})^{-1}\mathbf{C}_{a_t,s_t}\Sigma_{s_t}^{T}((-\mathbf{C}_{a_t,a_t})^{-1}\mathbf{C}_{a_t,s_t})^{T}.$$

We obtain, $\begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix} \sim N(\mu_{a_t,s_t}, \Sigma_{a_t,s_t})$ from this:

$$\mu_{a_t,s_t} = \begin{bmatrix} \mu_{a_t} \\ \mu_{s_t} \end{bmatrix},$$

$$\Sigma_{a_t,s_t} = \begin{bmatrix} \Sigma_{a_t} & (-\mathbf{C}_{a_t,a_t})^{-1}\mathbf{C}_{a_t,s_t}\Sigma_{s_t} \\ \Sigma_{s_t}^{T}((-\mathbf{C}_{a_t,a_t})^{-1}\mathbf{C}_{a_t,s_t})^{T} & \Sigma_{s_t} \end{bmatrix}.$$

This will be used to compute $\mathbb{E}\left[\sum_t \begin{bmatrix}\mathbf{a}_t \\ \mathbf{s}_t\end{bmatrix}\begin{bmatrix}\mathbf{a}_t \\ \mathbf{s}_t\end{bmatrix}^T\right]$ in every timestep. Similarly, it is easy to compute $N(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$:

$$\mu_{s_{t+1}} = \mathbf{A}\mu_{s_t} + \mathbf{B}\mu_{a_t},$$

$$\Sigma_{s_{t+1}} = \Sigma_{error} +$$

$$\begin{bmatrix}\mathbf{A} \\ \mathbf{B}\end{bmatrix}^T \begin{bmatrix} \Sigma_{s_t} & \Sigma_{s_t}^T((-\mathbf{C}_{a_t,a_t})^{-1}\mathbf{C}_{a_t,s_t})^T \\ (-\mathbf{C}_{a_t,a_t})^{-1}\mathbf{C}_{a_t,s_t}\Sigma_{s_t} & \Sigma_{a_t}\end{bmatrix}^T \begin{bmatrix}\mathbf{A} \\ \mathbf{B}\end{bmatrix}$$

$$= \Sigma_{error} + \mathbf{A}\Sigma_{s_t}^T\mathbf{A}^T + \mathbf{B}\Sigma_{a_t}^T\mathbf{B}^T + \mathbf{B}(-\mathbf{C}_{a_t,a_t})^{-1}\mathbf{C}_{a_t,s_t}\Sigma_{s_t}\mathbf{A}^T$$

$$+ \mathbf{A}\Sigma_{s_t}^T((-\mathbf{C}_{a_t,a_t})^{-1}\mathbf{C}_{a_t,s_t})^T\mathbf{B}^T.$$

We iteratively compute the distribution of of states, actions and state-action pairs and obtain the distribution of $\mathbf{s}_T$, which will be used to compute $\mathbb{E}[(\mathbf{s}_T - \mathbf{s}_G)(\mathbf{s}_T - \mathbf{s}_G)^T]$.

6.2.3 Bayesian Intention and Target Prediction

Given a unique learned trajectory model for each intention, I, the observed partial state sequence, $s_{1:t}$, the prior probability distribution over the duration of the total sequence, T, and the goal, G, we employ Bayes' rule to form the posterior probability distribution of the possible targets:

$$P(G, I, T|s_{1:t}) \propto \prod_{\tau=1}^{t} \pi(a_t|s_t, G, I, T)P(G, I, T). \tag{6.7}$$

The Bayesian formulation enables us to use a flexible prior distribution over the goals and intentions that reflects our initial belief given relevant environment information, such as pre-defined task type, object affordance and distance with objects. This allows for any method that estimates a strong prior distribution to be combined with the LQR model in order to improve the predictive accuracy.

6.2.4 Complexity Analysis

When the dynamics are linear, the recursive relation of Equations Equation 6.5 and Equation 6.6 have closed-form solutions that are quadratic functions and the action distribution is a conditional Gaussian distribution. This enables more efficient computation for large time horizons, T, with a large continuous state and action space than could be computed in the discrete case.

For trajectory inference, the proposed technique only requires $O(\mathrm{T})$ matrix updates. A strong advantage of this model is the fact that the matrix updates only need to be computed once when performing inference over sequences sharing the same time horizon and goal position. In order to further improve the efficiency of our computation we employ the Armadillo C++ linear algebra library for fast linear computation (64).

6.3 Experimental Setup

We employ the Cornell Activity Dataset (CAD-120) (37) in order to analyze and evaluate our predictive inverse linear-quadratic regulation model.

6.3.1 Cornell Activity Dataset

The dataset consists of 120 depth camera videos of long daily activities. There are ten high-level activities: making cereal, taking medicine, stacking objects, unstacking objects, microwaving food, picking objects, cleaning objects, taking food, arranging objects and having a meal. These high-level activities are then divided into ten sub-activities: reaching, moving, pouring, eating, drinking, opening, placing, closing, cleaning and null. Here, we regard different types of sub-activity as different intentions.

Consider the task of pouring cereal into a bowl. We decompose this high-level activity into the following sub-activities: reaching (cereal box), moving (cereal box above bowl), pouring (from cereal box to a bowl), moving (cereal box to table surface), placing (cereal box to table surface), null (moving hand back).

6.3.1.1 Modifications

The goal of the moving sub-activity is dependent on the sub-activity it precedes. For instance, if the next sub-activity is placing, then the goal of moving will be some location above the target surface of the placing sub-activity, but if the next sub-activity is eating, then the goal will be the area around the mouth. Therefore, we regard the moving sub-activity as the beginning portion of the latter sub-activity and combine them as one. We ignore the null sub-activity due to its lack of intention or goal.

In a similar fashion, we also separate the opening sub-activity into two different sub-activities, opening a microwave and opening a jar. The actions in these two tasks have very different movements and goals, and thus are considered as separate sub-activities. This re-

sults in nine sub-activity classifications, each of which we regard as a trajectory type with a target and intent.

6.3.1.2 Test set

The data is randomly divided into a training set and a test set. The test set consists of 10% of the demonstrated sequences with at least one sequence belonging to each sub-activity. The model is then trained on the training set and evaluated using the test set.

6.3.2 Model Fitting

6.3.2.1 Estimating the Quadratic Parameters

Our LQR model uses two separate parameter matrices, $\mathbf{M}$ and $\mathbf{M_f}$. We employ accelerated stochastic gradient descent with an adaptive (adagrad) learning rate and L_1 regularization (22; 68) on both parameter matrices simultaneously. This regularized approach prevents over-fitting of the parameter matrices for sub-activities with a low number of demonstrated trajectories.

6.3.3 Target and Intention Sampling

6.3.3.1 Intention Sampling

In the prediction task we separate the sub-activities into two categories, those that require an object in the hand and those that do not. This eliminates tasks from consideration that lack the required presence or absence of an object. For example, when the task is placing an object and there is no object in the hand.

6.3.3.2 Target Sampling

Target points are chosen for each sub-activity according to a Gaussian distribution of the observed endpoints of the active hand and the target object in the training data. The target object for the eating and drinking sub–activities are the head joint and the target for the placing sub-activity is the true endpoint of the active hand trajectory. The skeleton and object tracking data are then used with this observed distribution to compute the probable endpoints for the test trajectories. Since the placing sub-activity has no target object, points are randomly chosen on the placeable surfaces in the scene.

6.3.3.3 Segmentation and Duration Sampling

When deployed in real world applications the separation of intentions may not be clear. For this reason it is necessary to include a method to detect segmentation. The incorporation of T in Equation 6.7 allows for the direct inclusion of segmentation inference as an inherent feature of the proposed LQR method. This is due to the fact that goals, intentions, and duration are grouped together in each prediction.

The inclusion of duration is important as a poor prediction of the end of the current sub-activity will lead to an inaccurate starting step for the next sub-activity. For this reason it is necessary to achieve an accurate estimate of the length of each sub-activity.

Our LQR model requires that we specify a length for the total trajectory. Since we are only observing a part of this trajectory prior to forming a prediction, it is necessary to infer this duration. Here we use the observed average distance covered by each action for each sub-activity with the remaining distance to be covered from the current point and the sampled

target point in order to infer a probable duration. This is done using the following equation: $T = t + \text{dist}(s_t, s_g)/\text{avgdist}_a$. This value, T, represents the inferred segmentation for sub-activity a and target s_g.

6.3.4 Prior Distributions

In order to improve the predictive accuracy, it is often helpful to include a prior distribution on the sampled target points. This allows for the inclusion of additional knowledge about the problem domain to be added to the inference model. Here, the priors enable us to use the context information of the human robot interaction which is not included in trajectory data in our model. We use two prior distributions for this purpose which we join together into a single distribution due to the tuple nature of the sampled sub-activity and target points.

We evaluate the inverse LQR technique against 4 different probabilistic prediction methods, nearest target, nearest target from extended trajectory, a sub-activity sequence Naive Bayes distribution, and a combination of nearest target and the sub-activity sequence distribution which is also used as a prior distribution for our proposed LQR method.

6.3.4.1 Target Distance Prior

For the nearest target prior we use the Euclidean distance from the final state of the observed trajectory, s_t, and the sampled target, s_g, and a strength coefficient, α, in order to form a probability distribution over the possible target points using the following equation:

$$p(s_g|s_t) \propto e^{-\alpha \text{dist}(s_t, s_g)}. \tag{6.8}$$

6.3.4.2 Markov Intention Prior

Here we use a second order Markov model to form the probability distribution of a sub-activity given the previous two observed sub-activities in the sequence:

$$p(a_i|a_{i-1}, a_{i-2}) = \frac{N(a_i, a_{i-1}, a_{i-2})}{N(a_{i-1}, a_{i-2})}, \tag{6.9}$$

with $N(a_i, a_{i-1}, a_{i-2})$ being the number of occurrences of activity a_i being preceded by the two activities, a_{i-1} and a_{i-2} in the training dataset.

6.3.4.3 Combining for a Full Prior

The above two prior distributions are then combined together in order to form a distribution over both the sub-activity and the location of the sampled target points with a simple method: $p(s_g, a_i) \propto p(a_i|a_{i-1}, a_{i-2})p(s_g|s_t)$.

6.3.5 Prediction

6.3.5.1 Current Target

We calculate probabilistic predictions for each sampled target using our LQR method (Equation 6.7). Here the trajectory of the most active hand is observed and used to develop a likelihood model for each target.

6.3.5.2 Next Target

When predicting the probability of the next target (the target following the not-yet-reached current target), we use the calculated probability of the current target and the target points in order to develop a distribution for the next target point. Since there is no trajectory to

observe for the next target, this reduces to simply using the prior distribution of the next targets and the probability distribution of the current target given the current partial trajectory: $P(G_{i+1}, I_{i+1}, T_{i+1}|s_{1:t}) \propto P(G_i, I_i, T_i|s_{1:t})P(G_{i+1}, I_{i+1}, T_{i+1}|G_i, I_i, T_i)$.

6.3.5.3 Notes on Segmentation Prediction

In addition to the direct inference of segmentation, many sub-activities contain clear discernible goal behaviors. For instance, the placing sub-activity will end with the active hand releasing an object. If the object is tracked, which is necessary for target sampling, we can easily determine when this takes place and the hand moves away from the placed object. This is also the case for reaching, which ends when the hand is in contact with an object.

Utilizing these characteristics and accurate estimates of the duration yields impressive results for segmentation inference. We are able to achieve perfect segmentation detection once the hands move close enough to the inferred target to get an accurate estimate of the remaining duration, notably once the goal is reached or passed. This is also evident in the improved log-loss, which represents the probability of the true duration, goal, and intention.

6.4 Evaluation

6.4.1 Comparison Metrics

We evaluate the inverse LQR technique against the probability distributions obtained from calculating the three prior methods. These are the nearest target distribution, the Markov model on the sub-activity sequence, and the full prior distribution that is a combination of the two.

We compare these distributions against the LQR model with a flat distribution prior, and with each of the above respective distributions as priors.

6.4.2 Execution Time

In many tasks using predictive inference, it is important to minimize the execution time of the inference task. In this case, the proposed LQR method models, infers and makes predictions for the 63 trajectories of the test set in 1.052 seconds total with the parallel training of the parameters for all nine of the sub-activity types taking less than an hour. These execution times were collected on an Intel i7-3720QM CPU at 2.60GHz with 16 GB of RAM.

The use of linear dynamics in the models allows for this efficient computation which is extremely important when inferring over quickly executed sequences that require fast reactions using the predicted results. In addition, relative positions were not used in the state formulation since the time to compute a different relative sequence of states/actions for each possible target combination could result in non-real-time computation which is of key importance for human-robot-interaction. The speeds described above for the entire dataset translates to a target inference rate of over 1000Hz for real-time applications. This is a frequency that is many orders of magnitude greater than the discrete ATCRF method of inference (37).

6.4.3 Predictive Results

We compare the predictive accuracy of our method against the aforementioned techniques using the averaged mean log-loss (8; 57). This allows us to compare the likelihood of the demonstrated trajectory to the distance and activity measures previously discussed.

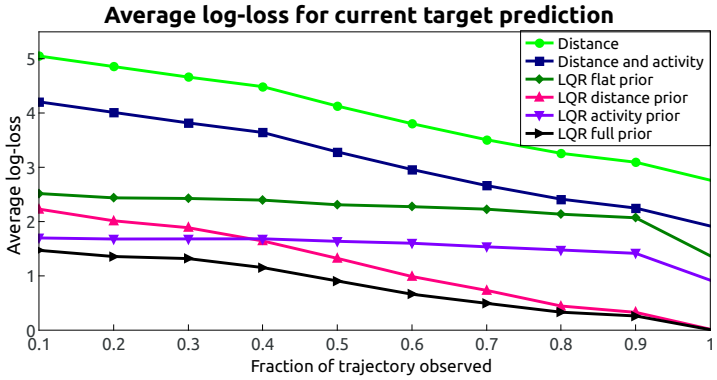

Figure 1: Average predictive loss of current target with partially observed trajectories. We compare the results of the distance and the distance-activity full prior with LQR.

As we show in Figure 1, the presented LQR method outperforms the other predictive techniques. This is mainly due to the incorporation of our sophisticated LQR likelihood model for the demonstrated sequence trajectories with the prior target distribution.

The activity prior is independent of the percentage of the sequence that is observed with an averaged mean log-loss for the activity model of 4.54. While this is not very good on its own, it does add a significant improvement to the LQR model when incorporated as a prior. Likewise, the nearest-target distance model gives an additional boost to the results of the LQR model when added as a prior with the combination of the two priors gave the best results.

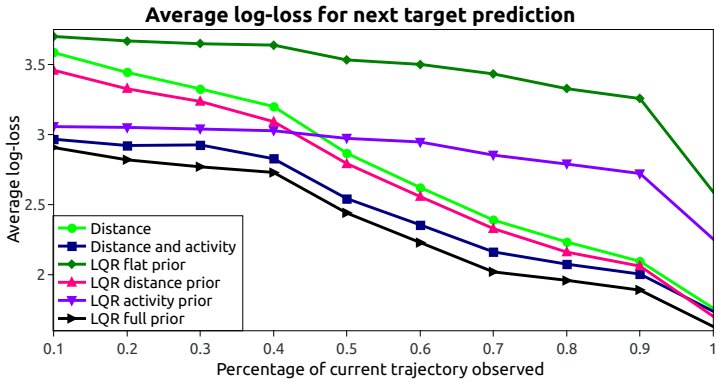

Figure 2: Average predictive loss of next target with partially observed trajectories. We compare the results of the distance and the distance-activity full prior with LQR.

The improvement seen in Figure 1 is then extended to the predictive results for the target of the next sub-activity in Figure 2. Each of the models uses the same technique to compute the next target distribution and then their respective methods for the computation of the current target distribution.

While the focus of this work is to improve the predictive log-loss of the model, it is helpful to use a classification accuracy in order to allow us to compare our results to prior techniques used on this data. In this case we choose the most probable target given the partially observed trajectory as the classified goal. Table I compares the results of the presented LQR technique with the ATCRF method (37) where macro precision and recall are the averages of precision and recall respectively for all sub-activities.

Results Comparison with Ground Truth Segmentation			
Method	Accuracy	Macro Precision	Macro Recall
LQR 20% sequence	80.9 $\pm$ 2.4	65.0 $\pm$ 3.1	77.3 $\pm$ 2.4
LQR 40% sequence	82.5 $\pm$ 3.2	73.4 $\pm$ 2.2	91.4 $\pm$ 0.6
LQR 60% sequence	84.1 $\pm$ 0,9	79.1 $\pm$ 2.5	94.2 $\pm$ 0.6
LQR 80% sequence	90.4 $\pm$ 0.4	87.5 $\pm$ 1.8	96.2 $\pm$ 0.3
LQR 100% sequence	100 $\pm$ 0.0	100 $\pm$ 0.0	100 $\pm$ 0.0
ATCRF 100% sequence	86.0 $\pm$ 0.9	84.2 $\pm$ 1.3	76.9 $\pm$ 2.6

Results Comparison without Ground Truth Segmentation			
Method	Accuracy	Macro Precision	Macro Recall
LQR 20% sequence	66.7 $\pm$ 3.9	50.1 $\pm$ 3.7	62.4 $\pm$ 3.8
LQR 40% sequence	69.8 $\pm$ 3.9	50.5 $\pm$ 4.4	56.7 $\pm$ 3.3
LQR 60% sequence	76.1 $\pm$ 2.7	72.2 $\pm$ 2.7	93.3 $\pm$ 0.5
LQR 80% sequence	77.8 $\pm$ 3.4	75.7 $\pm$ 2.9	93.5 $\pm$ 0.5
LQR 100% sequence	100 $\pm$ 0.0	100 $\pm$ 0.0	100 $\pm$ 0.0
ATCRF 100% sequence	68.2 $\pm$ 0.3	71.1 $\pm$ 1.9	62.2 $\pm$ 4.1

TABLE I: Accuracy and macro precision and recall with standard error for current activity detection when different percentages of the sequence are observed.

As we show in Table I, our LQR method has significantly improved upon the previous state of the art results with perfect classification accuracy when detecting the activity given the entire demonstrated trajectory. This is especially true when ground truth segmentation is not given. A likely reason for this is that obtaining perfect segmentation detection allows for us to use the true starting point for each sub-activity.

In addition, the LQR technique, given only 40% of the sequence obtains comparable results to the ATCRF model given the entire sequence. This is a significant result since the ability to predict the target and intention early on in a sequence is highly beneficial to many applications including human-robot interaction.

We note that the ATCRF results are on the unmodified CAD-120 dataset. As mentioned previously, we have merged the moving sub-activity with its succeeding sub-activity and separated the opening task into two different sub-activities. While this produces a more straight forward sampling task, it also makes the first half of the demonstrated sequences more ambiguous and segmentation more difficult. This is due to the similar dynamics of the moving sub-activity as compared to the other sub-activities. However, our results show that the proposed LQR method is robust enough to generate strong performance after only observing the first 20% of the sequence.

6.5 Discussion

In this section we have shown that incorporating the dynamics of a trajectory sequence into a predictive model elicits a significant improvement in the inference of target locations and activities for human task completion. We did this using linear quadratic regulation trained

with maximum entropy inverse optimal control and have shown that using linear dynamics in forming a model for task and target prediction improves intention recognition while providing efficient computation of the inferred probability distributions.

The combination of efficient inference with strong predictive results yields a very promising technique for any field that requires the predictive modeling of decision processes in real time. This is especially important in the area of human-robot collaboration where a robot may need to react to the inferred intentions of a human collaborator before they complete a task, which is an application that the authors plan to undertake in the near future.

While the results reported improve upon the conditional random field (ATCRF) technique (37), we feel the best results can be obtained by incorporating the discrete distribution learned using the ATCRF as an additional prior into the proposed LQR model. Any methods that improve the prior distribution (e.g., (67; 37)) will improve the log-loss since it is additive over the likelihood function and prior when taking the log of Equation 6.7. This combination of discrete and continuous learned models should return a strong predictive distribution of the targets and intentions by accounting for both the linear dynamics of the motion trajectories and the object affordances of the activity space during inference.

CHAPTER 7

WAYPOINT GUIDED INVERSE LINEAR QUADRATIC REGULATION

Part of this chapter was previously published in the International Joint Conference on Artificial Intelligence (http://www.ijcai.org/Abstract/15/266) (12).

7.1 Waypoint-based MaxEnt Inverse Linear-Quadratic Regulation

To better imitate the dynamics of continuous demonstrated trajectories, we construct a continuous MaxEnt IOC model that estimates a distribution of trajectories given a discrete path through our approximation graph. We learn a continuous path distribution through the discrete graphs that match the preferences elicited in the demonstrated trajectories. This allows for us to infer a path through the graph that correlates to the discretely inferred trajectory described previously while retaining the same continous motion patterns of the the demonstrated examples.

While the inverse linear quadratic regulation method described in the previous chapter performs very well in prediction tasks for standard linear motion, its limitations in more complex settings become prevalent when we consider the task on manipulating a robotic arm through a cluttered environment.

Figure 1: PR2 robot completing a task in a cluttered environment.

The many degrees of freedom (DoF) of typical robotic arms pose significant challenges for using inverse optimal control on manipulation tasks. IOC relies on solving the optimal control problem, but this is generally intractable and state-of-the-art planning methods provide no optimality guarantees. Furthermore, to robustly learn cost functions from non-optimal demonstrations requires the ability to reason about distributions over paths, rather than only computing the optimal one (81). For the specific case of systems with linear dynamics and quadratic cost functions, efficient inverse optimal control methods have been developed (46; 77; 52). However, these assumptions rarely hold in manipulation tasks. The desirable space of manipulation trajectories is often multi-modal and non-linear due to collision avoidance with discrete obstacles. Despite this, IOC methods have leveraged the special case of linear-quadratic systems by locally rationalizing demonstrated trajectories using a linear-quadratic approximation centered

around the demonstrated trajectory itself (46). Unfortunately, when a controller applies the resulting cost function to a new situation, the correct reference trajectory for linear-quadratic approximation is unknown. Other reference trajectories may produce inherently non-human-acceptable manipulation trajectories.

Previous IOC applications either employ discrete, but low-dimensional, decision process representations (2; 81) or make linear-quadratic assumptions about the state dynamics and cost function (46; 77). Thus, the linear-quadratic assumption is limited to obtaining cost function estimates that locally rationalize demonstrated trajectories around one of these modes (in practice: the mode defined by the demonstrated trajectory itself) (46). However, when the controller applies the resulting cost function to a new situation, the correct mode needed as a starting point is unknown. Other modes may produce inherently non-human-acceptable manipulation trajectories—even when the best of those is locally selected.

Motivated by the weaknesses of existing IOC methods for handling higher-dimensional control tasks and the limitations of linear quadratic control in the presence of discrete features such as obstacles, we propose a waypoint guided inverse linear quadratic regulation algorithm for learning continuous path distributions bounded by discrete paths. We demonstrate the effectiveness of this approach with several experiments conducted on a 7-DOF robotic arm.

In this section we extend the inverse linear quadratic regulation model presented in the previous chapter (52) in order to bind the continuous distribution around discrete feature representations, in the form of inferred discrete paths learned from demonstrated behavior. This

allows for a model to incorporate discrete features, such as obstacles, while avoiding the inherent inefficiency of planning full discrete trajectories.

7.2 Approach

We begin with the quadratic cost functions presented in the previous chapter via Equation 6.1 and Equation 6.2,

$$\mathrm{cost}(\mathbf{s}_t, \mathbf{a}_t) = \begin{bmatrix} \mathbf{a}_t \\ \\ \mathbf{s}_t \end{bmatrix}^T \mathbf{M} \begin{bmatrix} \mathbf{a}_t \\ \\ \mathbf{s}_t \end{bmatrix}, t < T,$$

$$\mathrm{cost}(\mathbf{s}_T) = (\mathbf{s}_T - \mathbf{s}_G)^T \mathbf{M_f} (\mathbf{s}_T - \mathbf{s}_G),$$

and add a new cost function that penalizes the state at time step t for deviating from some subgoal, or waypoint, temporally associated with step t,

$$\mathrm{cost}(\mathbf{s}_t) = (\mathbf{s}_t - \mathbf{s}_{w_t})^T \mathbf{M_w} (\mathbf{s}_t - \mathbf{s}_{w_t}). \tag{7.1}$$

This penalty functions in the same fashion as Equation 6.2 with a separate learned parameter matrix, $\mathbf{M_w}$, that is shared for each waypoint in the sequence. The sequence of waypoints can be generated via a discrete planner that infers a low fidelity trajectory through the problem space.

Similar to the previous chapter, in order to learn the appropriate parameters that model the demonstrated behavior we maximize the causal entropy under three constraints. Where a_t is the action at time step t, s_t is the state at time step t, s_G is the goal position, and s_{w_t} is the

position of the waypoint at t, if one exists. Here the waypoints function as intermediary goals where the task is to learn some cost function that penalizes states that deviate from the goal and waypoints. To optimize these parameters we aim to minimize the loss,

$$\begin{aligned}\mathbb{L} =& H(\mathbf{a}_{1:T}||\mathbf{s}_{1:T}) + M \cdot \left(\mathbb{E}_{\hat{p}} \left[\sum_t \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix} \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \right] - \mathbb{E}_{\tilde{p}} \left[\sum_t \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix} \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \right] \right) \\ &+ M_f \cdot \left(\mathbb{E}_{\hat{p}}[(\mathbf{s}_T - \mathbf{s}_G)(\mathbf{s}_T - \mathbf{s}_G)^T] - \mathbb{E}_{\tilde{p}}[(\mathbf{s}_T - \mathbf{s}_G)(\mathbf{s}_T - \mathbf{s}_G)^T] \right) \\ &+ M_w \cdot \left(\mathbb{E}_{\hat{p}}[\sum_{t \in t_w} (\mathbf{s}_t - \mathbf{s}_w)(\mathbf{s}_t - \mathbf{s}_w)^T] - \mathbb{E}_{\tilde{p}}[\sum_{t \in t_w} (\mathbf{s}_t - \mathbf{s}_w)(\mathbf{s}_t - \mathbf{s}_w)^T] \right)\end{aligned}$$

and maximize the entropy,

$$\max H(\mathbf{A}_{1:T}||\mathbf{S}_{1:T})$$

$$\text{s.t.} \quad \mathbb{E}_{\hat{p}} \left[\sum_t \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix} \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \right] = \mathbb{E}_{\tilde{p}} \left[\sum_t \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix} \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \right]$$

$$\mathbb{E}_{\hat{p}}[(\mathbf{s}_T - \mathbf{s}_G)(\mathbf{s}_T - \mathbf{s}_G)^T] = \mathbb{E}_{\tilde{p}}[(\mathbf{s}_T - \mathbf{s}_G)(\mathbf{s}_T - \mathbf{s}_G)^T]$$

$$\mathbb{E}_{\hat{p}} \left[\sum_{t \in t_w} (\mathbf{s}_t - \mathbf{s}_w)(\mathbf{s}_t - \mathbf{s}_w)^T \right] = \mathbb{E}_{\tilde{p}} \left[\sum_{t \in t_w} (\mathbf{s}_t - \mathbf{s}_w)(\mathbf{s}_t - \mathbf{s}_w)^T \right]$$

Solving the above optimization problem give us the following recursive relationship formed using the quadratic cost matrices M, M_f and M_w.

$$\hat{\pi}(\mathbf{a}_t|\mathbf{s}_t) = e^{Q(\mathbf{s}_t,\mathbf{a}_t)-V(\mathbf{s}_t)}$$

$$Q(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{\tau(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}[V(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)] + cost(\mathbf{s}_t, \mathbf{a}_t)$$

$$V(\mathbf{s}_t) = \begin{cases} \mathbb{I}_{t\in t_w}(\mathbf{s}_t - \mathbf{s}_{w_t})^T \mathbf{M}_w(\mathbf{s}_t - \mathbf{s}_{w_t}) + \text{softmax}(Q(\mathbf{s}_t, \mathbf{a}_t)), & t < T \\ (\mathbf{s}_t - \mathbf{s}_G)^T \mathbf{M}_f(\mathbf{s}_t - \mathbf{s}_G), & t = T \end{cases}$$

where $cost(\mathbf{s}_t, \mathbf{a}_t) = \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \boldsymbol{M} \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}$ and an indicator function, $\mathbb{I}$, signifying whether time step t is associated with a waypoint.

We can rewrite Q and V as quadratic functions in a similar manner to the inverse LQR method described in the previous chapter:

$$Q(\mathbf{s}_t, \mathbf{a}_t) = \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \begin{bmatrix} \mathbf{C}_{a,a} & \mathbf{C}_{a,s} \\ \mathbf{C}_{s,a} & \mathbf{C}_{s,s} \end{bmatrix} \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix} + \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \begin{bmatrix} \mathbf{F}_a \\ \mathbf{F}_s \end{bmatrix} + const(Q),$$

$$V(\mathbf{s}_t) = \mathbf{s}_t^T \mathbf{D} \mathbf{s}_t + \mathbf{s}_t^T \mathbf{G} + const(V),$$

which similarly leads to a set of recursive update rules:

$$\mathbf{D}_T = \mathbf{M}_f;$$

$$\mathbf{G}_T = -2\mathbf{M}_f\mathbf{s}_f$$

for t in $T-1...1$

$$\mathbf{C}_{a,a} = \mathbf{B}^T\mathbf{D}\mathbf{B} + \mathbf{M}_{a,a};$$

$$\mathbf{C}_{a,s} = \mathbf{B}^T\mathbf{D}\mathbf{A} + \mathbf{M}_{a,s};$$

$$\mathbf{C}_{s,a} = \mathbf{A}^T\mathbf{D}\mathbf{B} + \mathbf{M}_{s,a};$$

$$\mathbf{C}_{s,s} = \mathbf{A}^T\mathbf{D}\mathbf{A} + \mathbf{M}_{s,s};$$

$$\mathbf{F}_a = \mathbf{B}^T\mathbf{G};$$

$$\mathbf{F}_s = \mathbf{A}^T\mathbf{G};$$

$$\mathbf{D} = \mathbf{C}_{s,s} - \mathbf{C}_{a,s}^T\mathbf{C}_{a,a}^{-1}\mathbf{C}_{a,s} + \mathbb{I}_{t\in t_w}\mathbf{M}_w;$$

$$\mathbf{G} = \mathbf{F}_s - \mathbf{C}_{a,s}^T\mathbf{C}_{a,a}^{-1}\mathbf{F}_a - \mathbb{I}_{t\in t_w}(2\mathbf{M}_w\mathbf{s}_{w_t});$$

The matrices above can then be used to compute the policy at time step t:

$$\begin{aligned}
\hat{\pi}(\mathbf{a}_t|\mathbf{s}_t) &= e^{Q(\mathbf{s}_t,\mathbf{a}_t)-V(\mathbf{s}_t)} \\
&= \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \begin{bmatrix} \mathbf{C}_{a,a} & \mathbf{C}_{a,s} \\ \mathbf{C}_{s,a} & \mathbf{C}_{s,s} \end{bmatrix} \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix} + \begin{bmatrix} \mathbf{a}_t \\ \mathbf{s}_t \end{bmatrix}^T \begin{bmatrix} \mathbf{F}_a \\ \mathbf{F}_s \end{bmatrix} - \mathbf{s}_t^T \mathbf{D} \mathbf{s}_t - \mathbf{s}^T \mathbf{G} \\
&- \mathbf{s}_{w_t}^T \mathbf{M}_w \mathbf{s}_{w_t} - \frac{1}{2} \log |2\pi(-2C_{a,a})^{-1}| + \frac{1}{4} \mathbf{F}_a^T \mathbf{C}_{a,a}^{-1} \mathbf{F}_a
\end{aligned}$$

$$\propto N[\mathbf{a}_t | 2\mathbf{C}_{a,s}\mathbf{s}_t + \mathbf{F}_a, -2\mathbf{C}_{a,a}]$$

which is $N((-\mathbf{C}_{a,a})^{-1}\mathbf{C}_{a,s}\mathbf{s}_t - \frac{1}{2}\mathbf{C}_{a,a}^{-1}\mathbf{F}_a, (-2\mathbf{C}_{a,a})^{-1})$.

In addition to the policy at time step t, the state and action distributions can be computed from the matrices formed from the iterative update rule procedure. If $p(\mathbf{s_t}) = N(\mu_{s_t}, \Sigma_{s_t})$ and $p(\mathbf{a_t}) = N(\mu_{a_t}, \Sigma_{a_t})$ according to the Gaussian identities:

$$\mu_{a_t} = -\mathbf{C}_{a,a}^{-1}\mathbf{C}_{a,s}\mu_{s_t} - \frac{1}{2}\mathbf{C}_{a,a}^{-1}\mathbf{F}_a;$$

$$\Sigma_{a_t} = (-2\mathbf{C}_{a,a})^{-1} + (-\mathbf{C}_{a,a})^{-1}\mathbf{C}_{a,s}\Sigma_{s_t}^T((-\mathbf{C}_{a,a})^{-1}\mathbf{C}_{a,s})^T;$$

$$\mu_{a_t,s_t} = \begin{bmatrix} \mu_{a_t} \\ \\ \mu_{s_t} \end{bmatrix}$$

$$\Sigma_{a_t,s_t} = \begin{bmatrix} \Sigma_{a_t} & (-\mathbf{C}_{a,a})^{-1}\mathbf{C}_{a,s}\Sigma_{s_t} \\ \Sigma_{s_t}^T((-\mathbf{C}_{a,a})^{-1}\mathbf{C}_{a,s})^T & \Sigma_{s_t} \end{bmatrix};$$

$$\mu_{s_{t+1}} = \mathbf{A}\mu_{s_t} + \mathbf{B}\mu_{a_t};$$

$$\Sigma_{s_{t+1}} = \Sigma_{error} + \mathbf{A}\Sigma_{s_t}^T\mathbf{A}^T + \mathbf{B}\Sigma_{a_t}^T\mathbf{B}^T + \mathbf{B}(-\mathbf{C}_{a,a})^{-1}\mathbf{C}_{a,s}\Sigma_{s_t}\mathbf{A}^T + \mathbf{A}\Sigma_{s_t}^T((-\mathbf{C}_{a,a})^{-1}\mathbf{C}_{a,s})^T\mathbf{B}^T.$$

Sampling from this trajectory distribution generates smooth paths that incorporate discrete features while preserving the continuous motion dynamics of the demonstrated behavior.

The parameters of the above model can then be trained on a set of demonstrated trajectories and an associated set of discrete waypoints for each trajectory. This forms a convex optimization problem where the gradients are formed via expectation matching:

$$\nabla \mathrm{L}(\mathbf{M}) = \mathbb{E}_{\hat{\pi}} \left[\sum_{\mathrm{t}=1}^{\mathrm{T}-1} \begin{bmatrix} \mathbf{a}_{\mathrm{t}} \\ \mathbf{s}_{\mathrm{t}} \end{bmatrix} \begin{bmatrix} \mathbf{a}_{\mathrm{t}} \\ \mathbf{s}_{\mathrm{t}} \end{bmatrix}^{\mathsf{T}} \right] - \mathbb{E}_{\tilde{\pi}} \left[\sum_{\mathrm{t}=1}^{\mathrm{T}-1} \begin{bmatrix} \mathbf{a}_{\mathrm{t}} \\ \mathbf{s}_{\mathrm{t}} \end{bmatrix} \begin{bmatrix} \mathbf{a}_{\mathrm{t}} \\ \mathbf{s}_{\mathrm{t}} \end{bmatrix}^{\mathsf{T}} \right],$$

$$\nabla \mathrm{L}(\mathbf{M}_w) = \mathbb{E}_{\hat{\pi}} \left[\sum_{\mathrm{t} \in \mathrm{t}_w} (\mathbf{s}_{\mathrm{t}} - \mathbf{s}_w)(\mathbf{s}_{\mathrm{t}} - \mathbf{s}_w)^{\mathsf{T}} \right] - \mathbb{E}_{\tilde{\pi}} \left[\sum_{\mathrm{t} \in \mathrm{t}_w} (\mathbf{s}_{\mathrm{t}} - \mathbf{s}_w)(\mathbf{s}_{\mathrm{t}} - \mathbf{s}_w)^{\mathsf{T}} \right],$$

$$\nabla \mathrm{L}(\mathbf{M}_{\mathrm{f}}) = \mathbb{E}_{\hat{\pi}} \left[(\mathbf{s}_{\mathrm{T}} - \mathbf{s}_{\mathrm{G}})(\mathbf{s}_{\mathrm{T}} - \mathbf{s}_{\mathrm{G}})^{\mathsf{T}} \right] - \mathbb{E}_{\tilde{\pi}} \left[(\mathbf{s}_{\mathrm{T}} - \mathbf{s}_{\mathrm{G}})(\mathbf{s}_{\mathrm{T}} - \mathbf{s}_{\mathrm{G}})^{\mathsf{T}} \right].$$

7.3 Empirical Results

We evaluate our approach on three separate tasks with a 7-DOF Barrett Whole Arm Manipulator (BarrettWAM) shown in Figure 2.

Figure 2: Barrett Whole Arm Manipulator (BarrettWAM).

The tasks involve the arm moving in a specific manner through a workspace to reach a target object while avoiding obstacles:

$\mathcal{T}_1$. Approach the target object from the right side (from the arm's perspective)

$\mathcal{T}_2$. Approach the target object from the left side

$\mathcal{T}_3$. Carry a liquid filled can to a target destination without spilling

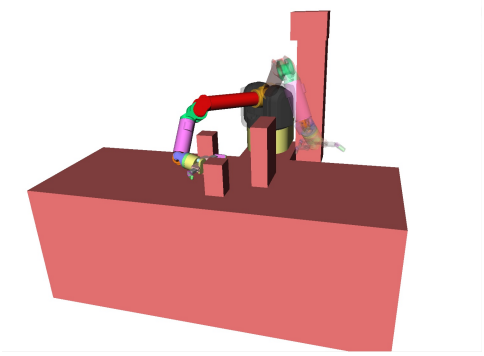

Figure 3: BarrettWAM and three objects with Start (transparent) and Goal (solid) configurations from the *Approach-Right* ($\mathcal{T}_1$) task. The target is the object closest to the end-effector.

For each task, we collected 16 demonstrations each from four different users with different start, goal and obstacle configurations. Half of the users had never used the robotic platform while the rest had some experience with it. In each case we recorded the joint angles of the demonstrated trajectories as well as the obstacle positions (segmented point clouds and

bounding boxes) via kinesthetic teaching. We use the OpenRAVE (19) virtual environment for testing our system as it provides good primitives for manipulator kinematics, geometry and visualization. Figure 3 shows an OpenRAVE render of a tabletop scene from one of our datasets.

7.3.1 Discrete Path Generation

In order to generate the waypoints needed to guide our LQR model we trained a discrete planner using graph-based inverse optimal control (12) which learns trajectory preferences through a coarse discrete approximation of the trajectory space and then "fine-tunes" the resulting path with a local trajectory optimizer based on the learned cost function.

Trajectories are projected through a coarse graph by computing the shortest path through the graph that most closely matches the demonstrated trajectory using a modified Dijkstra algorithm (20). Path nodes are chosen based on a mixed transition cost,

$$V_n = \min_{k=[i,...,g]}(||V_j - \xi_k||_2),$$
$$\text{for } \operatorname*{argmin}_{V_j \in \mathcal{N}(V_i)} \gamma||V_j - V_i||_2, \tag{7.2}$$

where V_i is the current node and $\min_{k=[i,...,g]}(||V_j - \xi_k||_2)$ represents the minimum distance between the closest (minimum cost chosen via $\operatorname*{argmin}_{V_j \in \mathcal{N}(V_i)} \gamma||V_j - V_i||_2$) neighboring node V_j and its closest point on the demonstrated trajectory ξ_k.

Maximum entropy inverse optimal control is then used (81) to optimize the cost functions of trajectories on the graph and local trajectory optimization is done on samples from the

maximum entropy distribution to infer smooth paths adhering to the coarse discrete structure of the graph. We refer to the literature for more details of ths method (12).

we trained our discrete planner using the following features:

1. Distance from robot to objects (min, average, target distance from end-effector, etc.);
2. Histogram over distance to objects;
3. Self-collision distances and corresponding histograms;
4. Histograms of position differences between end effector and target, right/left flags; and
5. Histograms over difference in end-effector orientation from start configuration

and additional features based on the start, goal and target object and a constant feature. In total, we have 95 discrete features for all of our tasks, a majority being integer or binary features and the rest individually scaled to be between 0-1.

7.3.2 Continuous state-action representation

For the continuous linear-quadratic regulation (LQR) representation we define the state, $\mathbf{s}_t = [\phi_t^1, ..., \phi_t^7, \dot{\phi}_t^1, ..., \dot{\phi}_t^7, \ddot{\phi}_t^1, ..., \ddot{\phi}_t^7, 1]^T$, and action, $\mathbf{a}_t = [\dot{\phi}_t^1, ..., \dot{\phi}_t^7]^T$, at time t where $(\phi^1, ..., \phi^7)$ denotes the joint angles, $(\dot{\phi}^1, ..., \dot{\phi}^7)$ the joint velocities, and $(\ddot{\phi}^1, ..., \ddot{\phi}^7)$ the joint accelerations. A unit constant is added to the state representation to incorporate linear features into the quadratic cost function formulation.

We use this representation, and the dynamics of the continuous demonstrated trajectories, to construct a waypoint guided LQR model that estimates a distribution of trajectories given a discrete path through the approximation graph generated via graph-based IOC (12). This

allows for us to infer a path through the graph that correlates to the discretely inferred trajectory while retaining the same continuous motion patterns of the the demonstrated examples.

7.3.3 Evaluation measures

We evaluate the trajectories sampled from the proposed approach against the demonstrated data. Inferred trajectories need to be collision-free while satisfying task-dependent metrics. For the *Can-Moving* task, we require that the end-effector's maximum deviation in pitch and roll from the start configuration be less than that of the demonstration. This ensures that a successful trajectory will not spill the liquid in the can. For the *Approach* tasks, we compute the percentage of the final 25% of the demonstration that stays on the desired side of the target requiring that the learned trajectory match or exceed this in order to be considered successful.

Additionally, we test on randomly generated scenes which each have a varying number of randomly generated box shaped objects (random number and size) in random configurations. The target object and the start and goal configuration of the robot are chosen randomly (subject to task constraints) as well. The metrics are similar for the random tests compared to the *Can-Moving* and *Approach* tasks, but fixed a priori.

7.3.4 Empirical Results

For each task, we trained our system using 70% of the demonstrations and test it on 30% of the demonstrations. In addition, we generated 20 random scenes per run for testing. In all tests, the discrete graph had 210,000 nodes with 15NN edges. We trained the LQR model on

Task		Collision-Free (%)	Completes Task (%)	Overall Success (%)
Approach-Right	Test	93	96	90
	Random	96	99	96
Approach-Left	Test	98	91	88
	Random	90	93	86
Can-Moving	Test	92	70	66
	Random	95	58	58

TABLE I: Learning performance on the withheld set (*Test*) and the randomly generated set (*Random*).

the fly using least-cost graph trajectories post learning and the training demonstrations. Table Table I shows the performance from our learned LQR model, averaged over six random trials.

7.3.4.1 Test Scenes

On the two *Approach* tasks the algorithm performs very well, avoiding obstacles and successfully completing the task on 89% of the scenes. For the *Can-Moving* task, the algorithm learns to avoid obstacles but fails to satisfy the task metric in nearly 30% of the scenes. One reason for reduced performance on the *Can-Moving* task is primarily due to the sparseness of the graph and projections in the discrete planner, leading to a poor representation of the small pitch and roll changes in the demonstrations. Instead, the projections themselves have larger variations (compared to the demonstrations); ultimately resulting in learned distributions that do not satisfy the maximum orientation difference metric. In this case the addition of the continuous motion dynamics into the model via waypoint guided inverse LQR resulted in a significant improvement in task completion (66% with LQR compared to 45% without).

7.3.4.2 Random Scenes

Additionally, we measure the performance on a set of randomly generated scenes using our previously-defined evaluation measures to quantify success. The proposed approach performs quite well on two of the three tasks it was tested on, successfully avoiding obstacles and completing the task in 91% of the random *Approach* scenes. For the *can-moving* task, the algorithm learns to avoid obstacles, but often fails to satisfy the task metric due to the limitations of the discrete planner for this task as described in the previous paragraph.

7.4 Discussion

This chapter addresses the limitations of the discrete approaches presented in Part I (eg. time efficiency) and proposes alternatives that can be used for real time tasks. For many tasks involving both discrete and continuous features it would be beneficial to incorporate the discrete distribution generated by the *SoftStar* algorithm of Part I as waypoints that guide the LQR distribution described in this chapter. This would allow for a model to be trained using accurate approximations of the discrete feature space while maintaining the dynamic linear features of a continuous model.

We applied the proposed waypoint guided inverse LQR method to three test scenarios with a 7DOF robot manipulator. The key strength of this approach is the ability to incorporate both discrete and continuous features efficiently into an IOC model. This is evident in the 21% improvement made in the *can-moving* task when waypoint guided LQR was added. Unfortunately it is sensitive to paths generated by the discrete planner and fails to produce strong results when the discrete path is very sparse.

CHAPTER 8

CONCLUSION

As our technology continues to evolve, so do the problems that we encounter. The challenge is that these problems come at increasing scales that require innovative solutions in order to be tackled efficiently. The key idea behind Inverse Optimal Control (IOC) is that we can learn to emulate how a human completes these complex tasks by modeling the observed decision process. In this thesis we presented algorithms that extend the state-of-the art in IOC in order to efficiently learn complex models of human behavior.

In Part I we presented two approaches to estimating the near-optimal path distribution through a weighted graph. In the first chapter we described a heuristic-guided policy iteration algorithm that builds on the Maximum Entropy Inverse Optimal Control framework (82). The following chapter addresses the limitations of sampling based policy iteration by proposing the *SoftStar* algorithm for probabilistic heuristic-guided search which estimates the same path distribution with added approximation guarantees.

We validated the presented methods on two complex decision problems; modeling handwriting trajectories of Latin characters, and modeling the ball-handling decision process of professional soccer teams. The results show that while heuristic-guided policy iteration performs well, the *SoftStar* algorithm clearly performs better due to the utilization of more informed updates generated from the approximate search leading to significantly faster convergence in optimizing the cost parameters.

Unfortunately, the computational cost of discretizing and searching a state-space can be problematic when a real-time solution is needed (eg. human-robot interaction). To address this issue we described two approaches in Part II for modeling continuous trajectories for prediction and planning. In the first section we presented an inverse LQR algorithm for predicting intent and forecasting trajectories for human-robot interaction. We achieved strong results that greatly improved on the previous state-of-the-art while maintaining the strong level of efficiency in terms of prediction time needed for such tasks. In the second section we extended this method to incorporate a discrete path that guides the continuous LQR distribution around discrete obstacles. This is important as many planning tasks are multi-modal and require knowledge of discrete features as well as the ability to model continuous motion dynamics.

Part II addresses the limitations of the discrete approaches presented in Part I (eg. time efficiency) and proposes alternatives that can be used for real time tasks. For many tasks involving both discrete and continuous features it would be beneficial to incorporate the discrete distribution generated by the *SoftStar* algorithm of Part I as waypoints that guide the LQR distribution described in the second chapter of part II. This would allow for a model to be trained using accurate approximations of the discrete feature space while maintaining the dynamic linear features of a continuous model further increasing the area of problems that can be efficiently solved.

8.1 Future Work

While we developed a number of different methods for handling large-scale inverse optimal control problems, there are always limitations and new challenges that can be addressed.

Modeling behavior in increasingly large scale decision processes presents challenges that continually require further research and exploration.

Adversarial methods have been applied to inverse optimal control problems where the demonstrated trajectories derive from a different decision process than the model being learned (16). These types of problem settings reduce the constraints of the model and allow for efficiently reduced decision processes to be applied to demonstrated behavior collected in a previously prohibitively complex decision process. Extending the methods developed in this thesis to take advantage of these ideas would lead to very efficient solutions to problems lying in previously intractable decision spaces.

Additionally, a current limitation of many of the presented approaches to inverse reinforcement learning is the use of a cost/reward function formed via a weighted linear combination of hand engineered features with a set of learnable parameters. Choosing these features requires low-level knowledge of the problem being solved which is not guaranteed in complex domains.

Recent work in deep reinforcement learning addressed this limitation by using neural networks to estimate state-action values Q a given raw visual input (49) using a variant of the Q-learning algorithm (76) and deep convolutional networks (43; 44). This eliminates the need for hand engineered features and learns a model free policy. The problem here is that this policy does not generalize to new tasks and the network must be retrained for each problem setting. Additionally, the use of epsilon greedy Q learning (76) results in the exploration of a large amount of sub-optimal sates before an efficient policy can be learned. An exploration

in merging these methods with the maximum entropy IOC framework could greatly extend the scope of the problems currently being addressed by both IOC and deep reinforcement learning.

The limitations of hand-engineered features and the requirement of modeling the decision process when learning demonstrated behavior are just a few examples of ways the methods presented in this thesis could be expanded. The challenges of modeling behavior in increasingly large scale decision processes and complex stat-spaces will require new and innovative solutions to be continually developed. The methods developed in this thesis provide a step towards this goal.

CITED LITERATURE

1. Abbeel, P. and Ng, A. Y.: Apprenticeship learning via inverse reinforcement learning. In Proceedings International Conference on Machine Learning, pages 1–8, 2004.

2. Abbeel, P. and Ng, A. Y.: Apprenticeship learning via inverse reinforcement learning. In Proceedings of the International Conference on Machine learning, page 1. ACM, 2004.

3. Aghasadeghi, N. and Bretl, T.: Maximum entropy inverse reinforcement learning in continuous state spaces with path integrals. In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, pages 1561–1566, Sept 2011.

4. Babes, M., Marivate, V., Subramanian, K., and Littman, M. L.: Apprenticeship learning about multiple intentions. In Proceedings International Conference on Machine Learning, pages 897–904, 2011.

5. Baker, C., Tenenbaum, J., and Saxe, R.: Goal inference as inverse planning. In Proceedings of the cognitive science society, 2007.

6. Baum, L. E.: An equality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. Inequalities, 3:1–8, 1972.

7. Beetz, M., Kirchlechner, B., and Lames, M.: Computerized real-time analysis of football games. IEEE Pervasive Computing, 4(3):33–39, July 2005.

8. Begleiter, R., El-yaniv, R., and Yona, G.: On prediction using variable order Markov models. journaltitle of Artificial Intelligence Research, 2004.

9. Bellman, R.: A Markovian decision process. journaltitle of Mathematics and Mechanics, 6:679–684, 1957.

10. Boularias, A., Kober, J., and Peters, J.: Relative entropy inverse reinforcement learning. In Proceedings of the International Conference on Artificial Intelligence and Statistics, pages 182–189, 2011.

11. Boyd, S., El Ghaoui, L., Feron, E., and Balakrishnan, V.: Linear matrix inequalities in system and control theory. SIAM, 15, 1994.

12. Byravan, A., Monfort, M., Ziebart, B., Boots, B., and Fox, D.: Graph-based inverse optimal control for robot manipulation. In International Joint Conference on Artificial Intelligence, 2015.

13. Byravan, A., Montfort, M., Ziebart, B., Boots, B., and Fox, D.: Layered hybrid inverse optimal control for learning robot manipulation from demonstration. In NIPS workshop on autonomous learning robots, 2014.

14. Castiello, U.: The neuroscience of grasping. Nature Reviews Neuroscience, 6(9):726–736, 2005.

15. Chajewska, U., Koller, D., and Ormoneit, D.: Learning an agent's utility function by observing behavior. In Proceedings of the International Conference on Machine Learning, pages 35–42, 2001.

16. Chen, X., Monfort, M., Ziebart, B. D., and Carr, P.: Adversarial inverse optimal control for general imitation learning losses and embodiment transfer. In UAI, 2016.

17. Daumé III, H. and Marcu, D.: Learning as search optimization: Approximate large margin methods for structured prediction. In Proceedings of the 22nd international conference on Machine learning, pages 169–176. ACM, 2005.

18. Dechter, R. and Pearl, J.: Generalized best-first search strategies and the optimality of a*. Journal of the ACM (JACM), 32(3):505–536, july 1985.

19. Diankov, R. and Kuffner, J.: Openrave: A planning architecture for autonomous robotics. Robotics Institute, Pittsburgh, PA, Technical Report, page 79, 2008.

20. Dijkstra, E. W.: A note on two problems in connexion with graphs. Numerische Mathematik, 1:269–271, 1959.

21. Duchi, J., Hazan, E., and Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. journaltitle of Machine Learning Research, July 2011.

22. Duchi, J., Hazan, E., and Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. journaltitle of Machine Learning Research, 12:2121–2159, July 2011.

23. Dudík, M. and Schapire, R. E.: Maximum entropy distribution estimation with generalized regularization. In Proceedings Computational Learning Theory, pages 123–138, 2006.

24. Feng, S., Manmatha, R., and McCallum, A.: Exploring the use of conditional random field models and hmms for historical handwritten document recognition. In Document Image Analysis for Libraries, 2006. DIAL'06. Second International Conference on, pages 30–37. IEEE, 2006.

25. Filipovych, R. and Ribeiro, E.: Combining models of pose and dynamics for human motion recognition. In International Symposium on Visual Computing, 2007.

26. Giffin, A. and Caticha, A.: Updating probabilities with data and moments. In Bayesian Inference and Maximum Entropy Methods in Science and Engineering, volume 954, pages 74–84, 2007.

27. Goodnow, J. J. and Levine, R. A.: the grammar of action: Sequence and syntax in children's copying. Cognitive Psychology, 4(1):82 – 98, 1973.

28. Green, P. J.: Reversible jump markov chain monte carlo computation and bayesian model determination. Biometrika, 82:711–732, 1995.

29. Hart, P., Nilsson, N., and Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics, 4:100–107, 1968.

30. Hauser, K.: Recognition, prediction, and planning for assisted teleoperation of freeform tasks. Autonomous Robots, 2013.

31. Henry, P., Vollmer, C., Ferris, B., and Fox, D.: Learning to navigate through crowded environments. In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pages 981–986. IEEE, 2010.

32. Huang, D.-A., massoud Farahman, A., Kitani, K. M., and Bagnell, J. A.: Approximate maxent inverse optimal control and its application for mental simulation of human interactions. In AAAI, 2015.

33. Kalman, R.: When is a linear control system optimal? Trans. ASME, J. Basic Engrg., 86:51–60, 1964.

34. Kidokoro, H., Kanda, T., Brscic, D., and Shiomi, M.: Will I bother here? A robot anticipating its influence on pedestrian walking comfort. In Human Robot Interaction, 2013.

35. Kim, K., Grundmann, M., Shamir, A., Matthews, I., Hodgins, J., and Essa, I.: Motion fields to predict play evolution in dynamic sport scenes. In In CVPR, 2010.

36. Kitani, K., Ziebart, B., Bagnell, J., and Hebert, M.: Activity forecastin. In Computer Vision European Conference on Computer Vision 2012, 2012.

37. Koppula, H. and Saxena, A.: Anticipating human activities using object affordances for reactive robotic response. In Robotics: Science ad Systems, 2013.

38. Koppula, H. S., Gupta, R., and Saxena, A.: Learning human activities and object affordances from rgb-d videos. International journaltitle on Robotic Research, 2013.

39. Kuderer, M., Kretzschmar, H., Sprunk, C. R. I., and Burgard, W.: Feature-based prediction of trajectories for socially compliant navigation. In Robotics: Science and Systems, 2012.

40. Lafferty, J., McCallum, A., and Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings International Conference on Machine Learning, pages 282–289, 2001.

41. Lake, B. M., Salakhutdinov, R., Gross, J., and Tenenbaum, J. B.: One shot learning of simple visual concepts. In Proceedings of the 33rd Annual Conference of the Cognitive Science Society, 2011.

42. Lake, B. M., Salakhutdinov, R., and Tenenbaum, J.: One-shot learning by inverting a compositional causal process. In NIPS, pages 2526–2534, 2013.

43. LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D.: Backpropagation applied to handwritten zip code recognition. Neural Comput., 1(4):541–551, December 1989.

44. Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, Nov 1998.

45. Lee, S. J. and Popović, Z.: Learning behavior styles with inverse reinforcement learning. In ACM SIGGRAPH 2010 Papers, SIGGRAPH '10. ACM, 2010.

46. Levine, S. and Koltun, V.: Continuous inverse optimal control with locally optimal examples. In ICML '12: Proceedings of the 29th International Conference on Machine Learning, 2012.

47. Lowerre, B. T. and Reddy, D. R.: The harpy speech understanding system. In Trends in Speech Recognition, pages 340–360. Prentice Hall, 1980.

48. Lucey, P., Bialkowski, A., Carr, P., Yue, Y., and Matthews, I.: how to get an open shot: Analyzing team movement in basketball using tracking data. In Workshop on Uncertainty in Artificial Intelligence, pages 55–63, 1988.

49. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D.: Human-level control through deep reinforcement learning. Nature, 518(7540):529–533, 02 2015.

50. Monfort, M., Lake, B. M., Lake, B. M., Ziebart, B., Lucey, P., and Tenenbaum, J.: Softstar: Heuristic-guided probabilistic inference. In Advances in Neural Information Processing Systems 28, eds, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, pages 2764–2772. Curran Associates, Inc., 2015.

51. Monfort, M., Lake, B. M., Ziebart, B. D., and Tenenbaum, J. B.: Predictive inverse optimal control in large decision processes via heuristic-based search. In ICML Workshop on Robot Learning, 2013.

52. Monfort, M., Liu, A., and Ziebart, B.: Intent prediction and trajectory forecasting via predictive inverse linear-quadratic regulation. In AAAI, 2015.

53. Neu, G. and Szepesvári, C.: Apprenticeship learning using inverse reinforcement learning and gradient methods. In Proceedings UAI, pages 295–302, 2007.

54. Ng, A. Y. and Russell, S.: Algorithms for inverse reinforcement learning. In Proceedings International Conference on Machine Learning, 2000.

55. Ng, A. Y. and Russell, S.: Algorithms for inverse reinforcement learning. In International Conference on Machine Learning, pages 663–670. Morgan Kaufmann, 2000.

56. Ng, A. Y. and Russell, S. J.: Algorithms for inverse reinforcement learning. In Proceedings of the International Conference on Machine Learning, 2000.

57. Nguyen, N. and Guo, Y.: Comparisons of sequence labeling algorithms and extensions. In International Conference on Machine Learning, 2007.

58. Ni, B., Wang, G., and Moulin, P.: Rgbd-hudaact: A color-depth video database for human daily activity recognition. In Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, pages 1147–1153, Nov 2011.

59. Pal, C., Sutton, C., and McCallum, A.: Sparse forward-backward using minimum divergence beams for fast training of conditional random fields. In Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on, volume 5, pages V–V. IEEE, 2006.

60. Pineau, J., Montemerlo, M., Pollack, M., Roy, N., and Thrun, S.: Towards robotic assistants in nursing homes: Challenges and results. Special issue on Socially Interactive Robots, Robotics and Autonomous Systems, 2003.

61. Pomerleau, D.: Alvinn: An autonomous land vehicle in a neural network. In Advances in Neural Information Processing Systems 1, 1989.

62. Ramachandran, D. and Amir, E.: Bayesian inverse reinforcement learning. In Proceedings International Joint Conferences on Artificial Intelligence, pages 2586–2591, 2007.

63. Ratliff, N., Bagnell, J. A., and Zinkevich, M.: Maximum margin planning. In Proceedings International Conference on Machine Learning, pages 729–736, 2006.

64. Sanderson, C.: Armadillo: An Open Source C++ Linear Algebra Library for Fast Prototyping and Computationally Intensive Experiments. Technical report, NICTA, September 2010.

65. Shapiro, S.: Encyclopedia of artificial intelligence. Number v. 1 in Encyclopedia of Artificial Intelligence. Wiley, 1992.

66. Strabala, K., Lee, M. K., Dragan, A., Forlizzi, J., Srinivasa, S., Cakmak, M., and Micelli, V.: Towards seamless human-robot handovers. journaltitle of Human-Robot Interaction, 2013.

67. Sung, J., Ponce, C., Selman, B., and Saxena, A.: Unstructured human activity detection from RGBD images. In International Conference on Robotic Automation, 2012.

68. Sutskever, I., Martens, J., Dahl, G. E., and Hinton, G. E.: On the importance of initialization and momentum in deep learning. In International Conference on Machine Learning, pages 1139–1147, 2013.

69. Sutton, R. S. and Barto, A. G.: Reinforcement Learning: An Introduction. The MIT Press, 1998.

70. Taga, G.: A model of the neuro-musculo-skeletal system for human locomotion. Biological Cybernetics, 73(2):97–111, 1995.

71. Toussaint, M.: Robot trajectory optimization using approximate inference. In Proceedings of the 26th Annual International Conference on Machine Learning, pages 1049–1056, 2009.

72. Toussaint, M. and Storkey, A.: Probabilistic inference for solving discrete and continuous state Markov Decision Processes. In Proceedings of International Conference on Machine learning, pages 945–952. ACM, 2006.

73. Trafton, J. G., Hiatt, L. M., Harrison, A. M., Tamborello, P., Khemlani, S. S., and Schultz, A. C.: Act-r/e: An embodied cognitive architecture for human-robot interaction. journaltitle of Human-Robot Interaction, 2013.

74. Trautman, P. and Krause, A.: Unfreezing the robot: Navigation in dense, interacting crowds. In International Conference on Intelligent Robots and Systems, 2010.

75. Vernaza, P. and Bagnell, D.: Efficient high dimensional maximum entropy modeling via symmetric partition functions. In Advances in Neural Information Processing Systems, pages 575–583, 2012.

76. Watkins, C. J. and Dayan, P.: Q-learning. Machine learning, 8(3):279–292, 1992.

77. Ziebart, B., Dey, A., and Bagnell, J. A.: Probabilistic pointing target prediction via inverse optimal control. In Proceedings of the 2012 ACM international conference on Intelligent User Interfaces, pages 1–10. ACM, 2012.

78. Ziebart, B. D., Bagnell, J. A., and Dey, A. K.: Modeling interaction via the principle of maximum causal entropy. In International Conference on Machine Learning, 2010.

79. Ziebart, B. D., Bagnell, J. A. D., and Dey, A.: The principle of maximum causal entropy for estimating interacting processes. IEEE Transactions on Information Theory, February 2013.

80. Ziebart, B. D., Dey, A. K., and Bagnell, J. A.: Fast planning for dynamic preferences. In Proceedings ICAPS, 2008.

81. Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K.: Maximum entropy inverse reinforcement learning. In AAAI, pages 1433–1438, 2008.

82. Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K.: Human behavior modeling with maximum entropy inverse optimal control. In Association for the Advancement of Artificial Intelligence Spring Symposium: Human Behavior Modeling, 2009.

83. Ziebart, B. D., Ratliff, N., Gallagher, G., Mertz, C. R. I., Peterson, K., Bagnell, J. A., Hebert, M., Dey, A. K., and Srinivasa, S.: Planning-based prediction for pedestrians. In International Conference on Intelligent Robots and Systems, 2009.

APPENDIX

COPYRIGHT POLICIES

A.1 Association for the Advancement of Artificial Intelligence (AAAI)

Authors who publish with this journal agree to the following terms:

1. Author(s) agree to transfer their copyrights in their article/paper to the Association for the Advancement of Artificial Intelligence (AAAI), in order to deal with future requests for reprints, translations, anthologies, reproductions, excerpts, and other publications. This grant will include, without limitation, the entire copyright in the article/paper in all countries of the world, including all renewals, extensions, and reversions thereof, whether such rights current exist or hereafter come into effect, and also the exclusive right to create electronic versions of the article/paper, to the extent that such right is not subsumed under copyright.

2. The author(s) warrants that they are the sole author and owner of the copyright in the above article/paper, except for those portions shown to be in quotations; that the article/paper is original throughout; and that the undersigned right to make the grants set forth above is complete and unencumbered.

3. The author(s) agree that if anyone brings any claim or action alleging facts that, if true, constitute a breach of any of the foregoing warranties, the author(s) will hold harmless and indemnify AAAI, their grantees, their licensees, and their distributors against any liability, whether under judgment, decree, or compromise, and any legal fees and expenses arising

APPENDIX (Continued)

out of that claim or actions, and the undersigned will cooperate fully in any defense AAAI may make to such claim or action. Moreover, the undersigned agrees to cooperate in any claim or other action seeking to protect or enforce any right the undersigned has granted to AAAI in the article/paper. If any such claim or action fails because of facts that constitute a breach of any of the foregoing warranties, the undersigned agrees to reimburse whomever brings such claim or action for expenses and attorneys fees incurred therein.

4. Author(s) retain all proprietary rights other than copyright (such as patent rights).

5. Author(s) may make personal reuse of all or portions of the above article/paper in other works of their own authorship.

6. Author(s) may reproduce, or have reproduced, their article/paper for the authors personal use, or for company use provided that AAAI copyright and the source are indicated, and that the copies are not used in a way that implies AAAI endorsement of a product or service of an employer, and that the copies per se are not offered for sale. The foregoing right shall not permit the posting of the article/paper in electronic or digital form on any computer network, except by the author or the authors employer, and then only on the authors or the employers own web page or ftp site. Such web page or ftp site, in addition to the aforementioned requirements of this Paragraph, must provide an electronic reference or link back to the AAAI electronic server, and shall not post other AAAI copyrighted materials not of the authors or the employers creation (including tables of contents with links to other papers) without AAAIs written permission.

APPENDIX (Continued)

7. Author(s) may make limited distribution of all or portions of their article/paper prior to publication.

8. In the case of work performed under U.S. Government contract, AAAI grants the U.S. Government royalty-free permission to reproduce all or portions of the above article/paper, and to authorize others to do so, for U.S. Government purposes.

9. In the event the above article/paper is not accepted and published by AAAI, or is withdrawn by the author(s) before acceptance by AAAI, this agreement becomes null and void.

A.2 Neural Information Proccessing Systems (NIPS)

All NIPS authors retain copyright of their work. You will need to sign a nonexclusive license giving the NIPS foundation permission to publish the work. Ultimately, however, you can do whatever you like with the content, including having the paper as a chapter of your thesis.

A.3 International Joint Conference on Artificial Intelligence (IJCAI)

You can publish any version of the paper published at IJCAI, IF a reference and a link to the copyright holder (IJCAI Organization) are clearly visible.