SEMANTICALLY GROUNDED LEARNING FROM UNSTRUCTURED DEMONSTRATIONS

A Dissertation Presented

by

SCOTT D. NIEKUM

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2013

School of Computer Science

SEMANTICALLY GROUNDED LEARNING FROM UNSTRUCTURED DEMONSTRATIONS

A Dissertation Presented

by

SCOTT D. NIEKUM

Approved as to style and content by:

Andrew G. Barto, Chair

Roderic A. Grupen, Member

Sridhar Mahadevan, Member

Michael L. Lavine, Member

Sonia Chernova, Member

Lori A. Clarke, Chair
School of Computer Science

for Edward Niekum, to whom I am forever grateful

ACKNOWLEDGMENTS

Just before starting my PhD, I asked several acquaintances who had already been through the process for advice. I was given ominous warnings, told horror stories, and was generally advised to finish as quickly as possible and not look back. All these years later, I'm happy and grateful to be able to say that, quite contrary to their expectations, I truly enjoyed the process of earning my PhD—I might even call it *fun*. I attribute this fact almost solely to the incredible group of people that I was surrounded by during graduate school.

First and foremost, my advisor Andy Barto was a joy to work with. In my early years as a student, he provided much-needed guidance and taught me a great deal about developing and communicating research ideas. In my later years, he entrusted me with the freedom to go wherever my research took me, giving me space to develop as an independent researcher, even when it meant wandering away from his typical areas of interest. Both methods of advising were critical to my success, and I am enormously grateful that Andy was willing to provide both. I can only hope that in the future I'm able to make even a fraction of the fundamental contribution that he has made during his distinguished career.

Sridhar Mahadevan, the co-director of the Autonomous Learning Lab, was also a source of inspiration during this thesis. His orientation toward heavy statistical machine learning was somewhat intimidating when I first arrived at UMass, but also inspired me to delve into it much more deeply than I may have otherwise—much to the benefit of my research. I am thankful for the rest of my committee as well: Rod Grupen, Michael Lavine, and Sonia Chernova have provided stimulating conversations, insights, mentoring, and mathematical advice whenever it was needed, and this dissertation wouldn't have been the same without them.

The most enjoyable parts of graduate school were undoubtably the experiences I had with the friends and collaborators I met at UMass. However, there is one person I feel compelled to thank above all others. I quite literally owe several years of my life to George Konidaris. From the day I entered the lab, he went far above and beyond the role of a senior lab member, helping to guide me through the program and avoid research pitfalls that could have cost me years of effort. After he moved on to a postdoc at MIT, he continued to be a mentor, collaborator, and friend, shaping my research career. Thank you George, it meant the world to me.

The rest of the Autonomous Learning Lab was fantastic as well. Phil Thomas provided years of inspiring conversation, a sounding board for ideas, friendship, and moral support. Will Dabney and Scott Kuindersma (and Jessica Kuindersma as well!) always helped to keep life fun, both inside and outside the lab, while also being excellent researchers and colleagues. Bruno Castro de Silva, Chris Vigorito, Andrew Stout, Yariv Levy, Tommy Boucher, CJ Carey, Jonathan Leahy, Tom Helmuth, Peter Krafft, Vimal Mathew, Bo Liu, Jie Chen, Chang Wang, Hoa Vu, Jeff Johns, Kim Ferguson, Armita Kaboli, Pippin Wolfe—thank you all.

I have many people to thank from the rest of the Computer Science department as well, and I apologize if I miss any of you: Gene Novark, Laura Sevilla, Bobby Simidchieva, Marc Cartright and Ilene Magpiong, Dirk Ruiken, Shiraj Sen, Jackie Field, Henry Field, Grant Sherrick and Charlotte Stanley, and Brandon McPhail. Also, a huge thank you to two unsung heroes in the department—Gwyn Mitchell for tirelessly dealing with problems with complicated NSF grants, postdoc paperwork, and travel funding, and Leeanne Leclerc for smoothing out countless instances of paperwork and bureaucracy.

I have been fortunate enough to have some extremely valuable collaborators from outside the department as well. Lee Spector from Hampshire college helped to jump-start my research during my first year and provided one of the most enjoyable research collaborations of my PhD. Sarah Osentoski, a former Autonomous Learning Lab alum, proved to be a great ally and colleague while at the Robert Bosch Research and Technology Center. Without Sarah, much of my robotics experience, funding opportunities, and upcoming postdoc would not have been possible.

Additionally, my mentors during my 6 month internship at Willow Garage—Bhaskara Marthi and Sachin Chitta—were great to work with. Willow Garage is one of the most exciting, fun places I have ever been, and I feel incredibly fortunate to have had the opportunity to spend so much time there. A significant portion of my thesis was completed at Willow, and I am very grateful for all the PR2 and ROS support I received while there, and for the incredible amount of research freedom I was given. Thanks to all the people that made being at Willow such a great experience: Aaron Blasdel, William Woodall, Mac Mason, Hilton Bristow, Adam Zimmerman, Dave Coleman, Stephen Brawner, Jonathan Mace, Chad Jenkins, Austin Hendrix, Ioan Sucan, Dirk Thomas, Jon Binney, Vincent Rabaud, Kaijen Hsaio, Matei Ciocarlie, Maya Cakmak, and many others I am sure I'm forgetting.

I would also like to extend my thanks to all my friends back in Pittsburgh. There are far too many to thank individually, but there are two I feel the need to single out. Jeff Railsback has been a constant source of positivity and encouragement in my life before and during grad school. He has provided couches and floors for me to sleep on countless times during my visits to the city over the years and has been one of the best all-around people to cheer me up in any circumstance. Erik Grieco has always felt like the brother I never had, almost from the first day we met. I am immensely grateful that our friendship has endured through these years spent apart, and can't wait to reunite during my postdoc in Pittsburgh.

My parents, Sandra and David Niekum, have been an unending source of love, support, and encouragement my entire life. As teachers, they always highly valued education, supported my academic interests from a young age, and never once discouraged my occasionally eccentric pursuits. It was okay that I insisted on dressing up as Albert Einstein for Halloween in second grade instead of a more normal kids' costume. I didn't even get in trouble for disassembling the family computer in grade school, having no idea how to put it back together. Instead, they gave me the chance to figure it out myself. As an adult, they have supported me in times of inspiration and in times when I was utterly adrift. Mom and Dad, thank you so much for weathering life's storms with me and never losing faith in me. I love you both very much and could never have done any of this without your support.

I must also thank my great-uncle Edward "Ep" Niekum, to whose memory this thesis is dedicated. Uncle Ep was part of the so-called "Greatest Generation"—a child of the Great Depression and a World War II veteran that served under Patton. He was a truck driver after the war and lived exceptionally frugally for his whole life. Thanks to him, I was able to attend the college of my choice without being crushed by debt afterwards, an uncommon privilege. He was also just a great man to be around and is dearly missed.

Finally, I can't imagine what this process would have been like without my loving partner, Rachael Singer. Meeting her during the first year of my PhD colored the rest of my grad school experience. All of my best memories of these years—celebrating milestones together, canoeing and swimming in the river every summer, getting snowed-in for days at a time—are all with her. She has had incredible patience with my often busy and frustrating lifestyle, always offering understanding and a good meal when I most need it. Rachael, I love you and I can't wait to see where life takes us.

This research was supported by the National Science Foundation under grant IIS-1208497 and the Air Force Office of Scientific Research under grant FA9550-08-1-0418. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation or the Air Force Office of Scientific Research.

ABSTRACT

SEMANTICALLY GROUNDED LEARNING FROM UNSTRUCTURED DEMONSTRATIONS

SEPTEMBER 2013

SCOTT D. NIEKUM

B.Sc., CARNEGIE MELLON UNIVERSITY

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Andrew G. Barto

Robots exhibit flexible behavior largely in proportion to their degree of semantic knowledge about the world. Such knowledge is often meticulously hand-coded for a narrow class of tasks, limiting the scope of possible robot competencies. Thus, the primary limiting factor of robot capabilities is often not the physical attributes of the robot, but the limited time and skill of expert programmers. One way to deal with the vast number of situations and environments that robots face outside the laboratory is to provide users with simple methods for programming robots that do not require the skill of an expert.

For this reason, learning from demonstration (LfD) has become a popular alternative to traditional robot programming methods, aiming to provide a natural mechanism for quickly teaching robots. By simply *showing* a robot how to perform a task, users can easily demonstrate new tasks as needed, without any special knowledge about the robot. Unfortunately, LfD often yields little semantic knowledge about the world, and thus lacks robust generalization capabilities, especially for complex, multi-step tasks.

To address this shortcoming of LfD, we present a series of algorithms that draw from recent advances in Bayesian nonparametric statistics and control theory to automatically detect and leverage *repeated structure* at multiple levels of abstraction in demonstration data. The discovery of repeated structure provides critical insights into task invariants, features of importance, high-level task structure, and appropriate skills for the task. This culminates in the discovery of semantically meaningful skills that are flexible and reusable, providing robust generalization and transfer in complex, multi-step robotic tasks. These algorithms are tested and evaluated using a PR2 mobile manipulator, showing success on several complex real-world tasks, such as furniture assembly.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Civilizations have dreamt for millennia of automata that serve their masters, from golems in ancient mythology, to the robots of modern science fiction. At present, autonomous robots are already used in settings such as industrial assembly, but can be dangerous, unwieldy, and exceptionally task-specific, falling short of a grander vision of robotics. The emerging field of *personal robotics* aims to fulfill the promise of intelligent robots that can safely serve, assist, and cooperate with humans in the home and workplace. Such robots could revolutionize the modern factory, help to take care of the disabled and elderly, or simply perform monotonous chores, possibly becoming as ubiquitous as the personal computer.

Robotic hardware has made great leaps in recent years, and control theory has advanced to the point that a wide range of behaviors can be programmed, given enough effort; robots now exist that can fold laundry [61], play catch [8], bake cookies [15], and play pool [78]. However, despite these impressive developments, general purpose personal robots that can operate competently in human environments are conspicuously missing. There are two central reasons for this. First, most robotic feats are performed in controlled conditions in the laboratory, whereas personal robots must be able to adaptively function in many different environments. Second, with few exceptions, robot behaviors are carefully hand-coded by experts to provide the robot with task-specific functionalities and semantic knowledge. While effective on a small scale, this is not a sustainable model for programming personal robots to perform the myriad of tasks desired by end-users.

One way to deal with the vast number of tasks and environments that personal robots must face outside the laboratory is to provide end-users with simple methods for programming robots that do not require the skill of an expert. Thus, in recent years, learning from demonstration (LfD) [4] has become a popular alternative to traditional robot programming methods, aiming to provide a natural mechanism for quickly teaching robots. By simply *showing* a robot how to perform a task, users can easily demonstrate new tasks as needed, without any specialized knowledge.

LfD has had many successes in teaching robots tennis swings [89], walking gaits [64], pick-and-place tasks [63], and even complex helicopter maneuvers [1]. However, demonstrations are often simply treated as trajectories to be mimicked, providing little semantic knowledge about the task or the environment. This, in turn, leads to brittle policies that often cannot generalize well to new situations—especially in the case of complex, multi-step tasks. The central theme of this dissertation is that demonstration data, in fact, contains *deep, exploitable structure* that can be discovered from a small number of examples and leveraged to provide robust generalization of complex tasks.

This leads us to ask: what information is necessary for strong generalization in complex robotic tasks? Generalization requires robots to adapt to new placements of objects in the world, identify and recover from possible contingencies, abstract away unimportant details, and recognize user intentions, among other things. All of these requirements can be summarized more simply as being able to identify what invariants must hold in a task, what is unimportant or allowed to vary, and why. The answers to these questions provide *explanatory power*, allowing the robot to explain the data it has observed and reason about what to do in new situations. However, to answer these questions, the robot requires a semantic understanding of the world—in our case, a collection of data that describes actions, objects, and their relationships.

We present a series of algorithms that look for *repeated structure* across multiple demonstrations to discover and exploit these vital task semantics.

The core of the presented algorithms are formed by Bayesian nonparametric models—models that do not have a fixed size, but instead infer an appropriate complexity in a fully Bayesian manner without overfitting the data or requiring model selection. These models are used to discover repeated structure in the demonstration data, identifying subgoals and primitive motions that best explain the demonstrations and that can be recognized across different demonstrations and tasks. This process converts noisy, continuous demonstrations into a simpler, coherent discrete representation. This discrete representation can then be leveraged to find additional structure, such as appropriate coordinate frames for actions, task-level sequencing information, and higher-level skills that are *semantically grounded.* Finally, this collection of data is combined to construct robust controllers that use an understanding of the world to adaptively perform complex, multi-step tasks. These algorithms are evaluated on several difficult real-world tasks, including furniture assembly, using a PR2 mobile manipulator.

1.1 Contributions and Chapter Outline

- **Chapter 2: Background and Related Work.** This chapter provides background material on the roots of time-series analysis, Bayesian nonparametric models, and learning from demonstration. Prior work and alternate approaches to LfD, subgoal identification, and skill discovery are discussed. This discussion is limited to topics that are relevant to this dissertation as a whole; background topics that are more chapter-specific are contained in the relevant chapter.

- **Chapter 3: Portable Subgoal Discovery.** One of the simplest types of user "demonstrations" are those in which the user specifies domain-specific events that are task-critical, but does not need to (and may not be able to) demonstrate *how* these events can be brought about. We describe a method for discovering useful, reusable skills in this setting, using a Bayesian nonparametric clustering algorithm and reinforcement learning. The agent collects data from experience and automatically discovers an appropriate number of subgoals that describe the sets of conditions that can bring about critical events. Skills, or *options*, are then learned to accomplish and sequence these subgoals. This method of skill discovery places very little responsibility on the user, but it is highly data intensive, making it inappropriate for use on physical robots. This difficulty motivates the contribution of the following chapters.

- **Chapter 4: Learning from Unstructured Demonstrations.** Fortunately, for tasks that can be demonstrated by an expert, a great deal of additional structure can be leveraged to make skill discovery and learning more data efficient. We present an algorithm that uses a Bayesian nonparametric model to segment unstructured demonstrations into motion categories, or skills, that can be recognized across demonstrations and tasks, maximally leveraging available data. This, combined with a coordinate frame detection algorithm, allows the robot to learn a multi-step task from a small number of demonstrations and generalize it to novel situations.

- **Chapter 5: Incremental Semantically Grounded Skill Discovery.** Next, we show how this approach can be extended to handle complex tasks that may have multiple valid execution paths, contingencies that may arise during execution, or that exhibit partial observability and/or perceptual aliasing. We introduce a novel method to further subdivide discovered motion categories into semantically grounded states in a finite-state representation of the task, enabling intelligent, adaptive replay. Performance can then be incrementally improved through interactive corrections that provide additional data where they are most needed. Together, this allows for intelligent discovery and sequencing of semantically grounded skills to create flexible, adaptive behavior that can be improved through natural interactions with the robot.

- **Chapter 6: Conclusions and Future Work.** We conclude by summarizing the work presented in this dissertation and highlighting opportunities for future research in learning from demonstration.

CHAPTER 2
BACKGROUND AND RELATED WORK

This chapter surveys of several areas of artificial intelligence and machine learning that provide a context for the contributions of this thesis. First, related work in skill learning is discussed within the context of learning from demonstration and reinforcement learning. Next, the Hidden Markov Model is introduced as a Bayesian approach to time-series analysis, which will be used when analyzing time-series demonstration data in Chapters 4 and 5. Finally, nonparametric Bayesian methods are introduced to provide a principled way to overcome the problem of model selection that often accompanies models with finite parameterizations, such as the Hidden Markov Model. This discussion is limited to topics that are relevant to this dissertation as a whole; background topics that are more chapter-specific are contained in the relevant chapter.

2.1 Skill Learning

Many approaches have been proposed to allow agents to learn skills that ease the learning of complex tasks and that are reusable in new situations. The majority of these methods fall into two major categories: learning from demonstration methods and reinforcement learning methods. This section provides an overview of these approaches to skill discovery and learning.

2.1.1 Learning from Demonstration

Learning from demonstration (LfD) [4, 11] is an approach to robot programming in which users demonstrate desired skills to a robot. Ideally, nothing is required of the user beyond the ability to demonstrate the task in a way that the robot can interpret. Example demonstration trajectories are typically represented as time-series sequences of state-action pairs that are recorded during the teacher's demonstration. The set of examples collected from the teacher is then often used to learn a policy (a state to action mapping) or to infer other useful information about the task that allows for generalization beyond the given demonstrations.

A variety of approaches have been proposed for LfD, including supervised learning [5, 18, 20, 38, 27, 2], reinforcement learning [93, 1, 109, 51], and behavior based approaches [71]. However, this work has generally been limited to tasks that can be represented with a single monolithic policy or that are broken up into subtasks by hand. In a recent example, Pastor et al. [78] use Dynamic Movement Primitives (DMPs) [45] to acquire single motor skills from structured demonstrations of a complex billiards shot. In their framework, multiple imperfect demonstrations of a skill are used to learn the initial parameters of a DMP controller, which is then improved using reinforcement learning. In addition, the statistics collected from the examples are used to predict the outcome of new executions of the same skill, to allow early termination if failure seems likely.

While many approaches enable the learning of a single policy or skill from data, some approaches perform automatic segmentation of the demonstrations into multiple skills. Jenkins and Matarić introduced Spatio-Temporal Isomap in order to find the underlying low-dimensional manifolds within a set of demonstrated data [47, 48]. This work extends the dimensionality reduction technique Isomap to include temporal information and allows the discovery of repeated motion primitives. However, segmentation in this model is performed with a heuristic. Dixon and Khosla [26] demonstrate that generalizable motions can be parameterized as linear dynamical systems. This algorithm also uses heuristic segmentation and cannot recognize repeated instances of skills.

Gienger et al. [35] segment skills based on co-movement between the demonstrator's hand and objects in the world and automatically find appropriate task-space abstractions for each skill. Their method can generalize skills by identifying task frames of reference, but it cannot describe skills like gestures or actions in which the relevant object does not move with the hand. Rather than explicitly separate skills into separate policies, work by Cederborg et al. [19] represents a flexible number of skills as a single policy using the Incremental Local Online Gaussian Mixture Regression algorithm. These skills are implicitly separated through the use of a local Gaussian mixture representation, where each mixture is localized in a different part of the state space.

Other recent work has examined using principled statistical techniques to segment example trajectories into multiple skills. Grollman and Jenkins [40] introduce the Realtime Overlapping Gaussian Expert Regression (ROGER) model to estimate the number of subtasks and their policies in a way that avoids *perceptual aliasing*, a condition in which perceptual information alone is not sufficient to choose the correct next action. Butterfield et al. [17] extend the Hierarchical Dirichlet Processes Hidden Markov Model (HDP-HMM) to handle perceptual aliasing and automatically discover an appropriate number of skills. Although we introduce a Bayesian mechanism to parse demonstration trajectories in this thesis, rather than inferring policies directly, we discover repeated dynamical systems which are considerably simpler to model than policies and generally require less data for inference.

The CST algorithm [52, 51] uses an online changepoint detection method to segment example trajectories and then merges the resulting chains of skills into a skill tree. This approach simultaneously segments the trajectories and discovers abstractions, but cannot recognize repeated skills to assist with the segmentation process. Kulic et al. [56] demonstrate an online method that can recognize repeated motion primitives to improve segmentation as additional data is collected by assuming that data points from the same primitive are generated by the same underlying distribution. Ciappa and Peters [21] model repeated skills as being generated by one of a set of possible hidden trajectories, which is rescaled and noisy. To guide segmentation, they define an upper bound on the number of possible skills and explicitly constrain segment lengths.

Several less traditional approaches to LfD exist as well. One approach by Rozo et al. [88] doesn't consider trajectory learning at all, but instead learns to cooperate with a human by learning appropriate stiffness and impedance policies from haptic information gathered from two-person task demonstrations. Kjellstrom and Kragic [50] eschew traditional policy learning, and instead use visual data to watch the demonstrator's hand to learn the affordances of objects in the environment, leading to a notion of object-centric skills. Ekvall and Kragic [29] use multiple examples of a task to learn task constraints and partial orderings of primitive actions so that a planner can be used to reproduce the task.

2.1.2 Inverse Reinforcement Learning

A special case of learning from demonstration is that of inverse reinforcement learning (IRL) [70], in which the agent tries to infer an appropriate reward or cost function from demonstrations. Thus, rather than try to infer a policy directly from the demonstrations, the inferred cost function allows the agent to learn and improve a policy from experience to complete the task implied by the cost function. This is typically done via policy search or reinforcement learning methods.

IRL techniques typically model the problem as an Markov Decision Process (MDP) and require an accurate model of the environment [1, 85, 67], but two recent methods have been proposed to circumvent this requirement by creating local control models [97], and by using an approach based on KL-divergence [16], respectively. Maximum entropy methods have also been suggested as a way to deal with ambiguity in a principled probabilistic manner [109]. As with other LfD methods, IRL can be used to teach robots any number of individual skills, but to the best of our knowledge has not yet been used to automatically discover multiple skills within demonstration data.

2.1.3 Reinforcement Learning and Subgoal Discovery

RL has a rich history of successes in learning control policies from interaction with the environment, ranging from world-class backgammon play [101], to autonomous inverted helicopter flight [68]. However, RL has only had limited success in general robotics applications. RL algorithms can require a large amount of exploration before a reasonable policy is found, and collecting experience on physical robots can be both time-consuming and expensive. Worse, in high-dimensional state spaces implied by robots with many degrees of freedom, commonly used random exploration strategies are not tractable (when starting from a random policy) and can also be dangerous to the robot, objects, and people in the environment. Continuous state and action spaces that are often present in robotic domains also pose challenges to traditional RL algorithms. Function approximation is commonly used to represent policies in continuous state spaces, but typical RL methods do not provide any of the stability guarantees that control theoretic approaches do. Continuous action spaces can also make the traditional notion of policy optimization in RL very difficult. One notable method for learning in continuous action spaces is the Natural Actor-Critic [81].

The difficulties that RL faces in real-world, continuous, high-dimensional problems can often be mitigated by breaking tasks into simpler subgoals that can each be accomplished with a single skill. Much previous work has investigated ways to automatically discover subgoals in RL tasks to be used as the goals of *options*. The options framework [95] models skills as temporally extended actions that can be invoked like primitive actions. An option o consists of an *option policy* $\pi_o : S \times A \rightarrow [0, 1]$, giving the probability of taking action a in state s, an *initiation set* $\mathcal{I}_o \subseteq S$, giving the set of states from which the option can be invoked, and a *termination condition* $\beta_o : S \rightarrow [0,1]$, giving the probability that option execution will terminate upon reaching state s.

However, most subgoal discovery algorithms work only in discrete state spaces and require a large amount of data, limiting the practical applicability of this work, especially in robotic domains where data is a scarce resource. Skill chaining [55] is a notable exception that can automatically discover subgoal regions in continuous state spaces. More fundamentally, many skills of interest such as walking or throwing a ball cannot be fully described by a skill objective that is described only as an endpoint in state space. Instead they are best characterized by dynamical properties that change with time.

The simplest subgoal discovery algorithms analyze reward statistics or state visitation frequencies to discover subgoal states [25]. Graph-based algorithms [92, 59] search for "bottleneck" states on state transition graphs via clustering and other types of analysis. Algorithms based on intrinsic motivation have included novelty metrics [91] and hand-coded salience functions [7]. Skill chaining [55] discovers subgoals by 'chaining' together options, in which the termination set of one option is the empirically determined initiation set of the next option in the chain. HASSLE [6] clusters similar regions of state space to identify single-task subgoals. Reward function search [72] has also been shown to extract features that could be considered subgoals, though to the best of our knowledge, no one has tried analyzing such reward functions to discover subgoal states.

Other algorithms analyze tasks to create skills directly, rather than search for subgoals. The VISA algorithm [49] creates skills to control factored state variables in tasks with sparse causal graphs, and one extension to this method adds an active learning component for efficient exploration [105]. PolicyBlocks [82] looks for policy similarities that can be used as templates for skills. The SKILLS algorithm [103] attempts to minimize description length of policies while preserving a performance metric. However, these methods only exhibit transfer to identical state spaces and often rely on discrete state representations. Related work has also used clustering to determine which of a set of MDPs an agent is currently facing, but does not address the need for skills within a single MDP [108].

2.2 Time-Series Analysis and the Hidden Markov Model

Time-series data, such as robot task demonstrations, present unique challenges for analysis, since observations are temporally correlated. Clearly, time-series data are not independent and identically distributed (nor are they exchangeable), but weaker assumptions can often be made about the data to make inference tractable.

Define a sequence to be a *first-order Markov chain* if:

$$p(x_n|x_1, x_2, \ldots, x_{n-1}) = p(x_n|x_{n-1}). \tag{2.1}$$

In other words, given the previous observation x_{n-1}, an observation x_n is conditionally independent of all other previous observations. To capture longer-range interactions, this concept can be extended to higher orders, such that an observation is dependent on the previous M observations:

$$p(x_n|x_1, x_2, \ldots, x_{n-1}) = p(x_n|x_{n-1}, \ldots, x_{n-M}). \tag{2.2}$$

One way to tractably model time-series data is through the use of a *state space model*, in which each observation x_i has a corresponding latent variable or *hidden state* z_i associated with it. The latent variables $z_1, \ldots, z_n$ form a Markov chain and *emit* the observations $x_1, \ldots, x_n$ based on conditional distributions of the form $p(x|z)$. Figure 2.1 shows the graphical representation of a state space model.

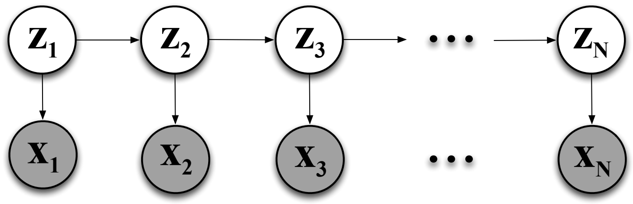


Figure 2.1. A state space model, or standard HMM when the latent variables $\mathbf{z}$ are discrete.

When the latent variables $\mathbf{z}$ in a state space model are discrete[1], we obtain the standard Hidden Markov Model (HMM). The standard HMM is defined by the number of states K that the latent variables can take on, a $K \times K$ transition probability matrix $\boldsymbol{\pi}$ (with rows $\boldsymbol{\pi}_k$) that describes the probabilities $p(z_i|z_{i-1})$, and a parameterized distribution[2] $F(\cdot)$ that describes the conditional probabilities $p(x_i|z_i)$. The generative model for an HMM can be written as:

$$
\begin{aligned}
z_i &\sim \boldsymbol{\pi}_{z_{i-1}} \\
x_i &\sim F(\theta_{z_i}),
\end{aligned}
$$

where θ_{z_i} is a parameter vector associated with state z_i, and "$\sim$" can be read as "drawn from" or "distributed as". In other words, the HMM can describe time series data with a mixture model in which the latent mixture component indices have a temporal relationship as a first-order Markov chain.

One drawback of the standard HMM is that the observation x_i is conditionally independent of any other observation x_j, given the generating hidden state z_i. This

[1]The term "Hidden Markov Model" is typically used to refer to a model with discrete latent variables. When latent variables are continuous, a discrete transition matrix can no longer be used to describe transitions, instead requiring formulations like linear-Gaussian systems. These models are sometimes referred to by different names in the literature, rather than as an HMM.

[2]Technically, each hidden state z can have its own unique parameterized distribution $F_z(\cdot)$, but in practice all hidden state emission distributions usually share a common form, such as a Gaussian distribution. However, each hidden state still has a unique set of parameters θ_z for this distribution.

independence assumption is clearly not well founded for much time-series data, such as robot demonstrations, in which the observations, as well as the hidden states, have temporal dependencies. The autoregressive HMM (AR-HMM) addresses this by adding links between successive observations, forming a Markov chain, as shown in Figure 2.2. This can be extended to a M^{th} order AR-HMM, as shown in Figure 2.3.

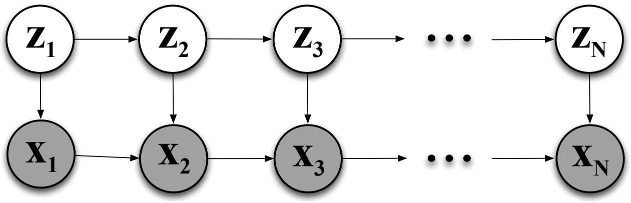


Figure 2.2. A first-order autoregressive HMM.

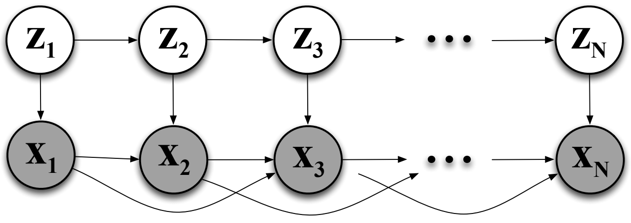


Figure 2.3. Additional links can be added to make an M^{th}-order autoregressive HMM. This example shows a second-order AR-HMM.

AR-HMMs face a problem in cases where the observations are not discrete—a simple transition matrix cannot be used to describe the probabilities $p(x_i|z_i, x_{i-1}, \ldots, x_{i-M})$. For example, demonstration data is often comprised of continuously valued state-action pairs representing robot joint poses and actuations. In this case the conditional probability density function over x_i must be able to be written in terms of a continuous function of its predecessors. For example, a linear dynamical system governing the conditional distribution can be used, such that:

$$p(x_i|z_i, x_{i-1}, \ldots, x_{i-M}) = \sum_{j=1}^{M} A_{j,z_i} x_{i-j} + \mathbf{e}_i(z_i),$$

for transition matrices A_{j,z_i} and covariance terms $\mathbf{e}_i(z_i)$. In this way, time-series data can be flexibly modeled as a mixture of linear dynamical systems.

2.3 Nonparametric Bayesian Methods

Graphical models and Bayesian inference have seen great success in artificial intelligence and machine learning applications spanning many fields including natural language processing [14], computer vision [84], social science [65], genetics [10], and medicine [57]. The Bayesian paradigm provides principled mechanisms to allow the specification of prior beliefs, model dependencies between variables, and perform efficient inference. However, one persistent difficulty in Bayesian inference is that of *model selection*. Often, it is not known *a priori* how complex a model must be to capture all the important structure of a dataset without overfitting, making it difficult to even provide a reasonable prior over such models. A classic example of this is choosing the polynomial degree in polynomial regression; too low of a degree will miss important characteristics of the data, while too high of a degree will begin to fit noise. This problem is also commonly found when trying to determine the appropriate number of components in a mixture model or the number of hidden states in an HMM.

Many techniques have been developed to select amongst models with fixed, finite parameterizations, including cross validation, Bayesian information criterion, and maximization of model evidence. However, many of these methods rely on heuristics, approximations, or the specification of a prior over model complexity, which may not be practical to specify for complex data. By contrast, Bayesian nonparametric methods sidestep having to explicitly perform model comparison by using an infinite parameterization that can determine an appropriate model complexity directly from data in a fully Bayesian manner. The next section will discuss how such parameterizations are possible. These infinite, nonparametric representations will be used in Chapter 3 to automatically select an appropriate number of data clusters, representing candidate skill goals, and in Chapters 4 and 5 to determine an appropriate number of dynamical systems that adequately describe the observed demonstration data.

2.3.1 De Finetti's Theorem

A common simplifying assumption in statistics is that of *independence*, in which the joint probability of data can be expressed as the product of the probabilities of each individual data point:

$$p(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} p(x_i). \tag{2.3}$$

A somewhat weaker assumption that nonparametric Bayesian methods leverage is that of *exchangeability*. A finite sequence of random variables is considered to be finitely exchangeable if every possible permutation of the random variables has an identical joint distribution, making the order in which data arrives irrelevant. An infinite sequence is considered infinitely exchangeable if every finite subset is finitely exchangeable. It can be seen from this that independence implies exchangeability, but that exchangeability does not necessarily imply independence.

De Finetti's theorem [23] states that a sequence $x_1, x_2, \ldots$ of binary random variables is infinitely exchangeable if and only if there exists a random variable θ with cumulative distribution function Q on $[0, 1]$, such that for all n:

$$p(x_1, x_2, \ldots, x_n) = \int_0^1 \prod_{i=1}^{n} \theta^{x_i}(1-\theta)^{1-x_i} dQ(\theta), \tag{2.4}$$

or equivalently:

$$p(x_1, x_2, \ldots, x_n) = \int_0^1 p(x_1, x_2, \ldots, x_n|\theta) dQ(\theta), \tag{2.5}$$

where $Q(\theta)$ is typically (but not required to be) well-behaved, such that $dQ(\theta) = Q'(\theta)d\theta = p(\theta)d\theta$. De Finetti only proved this for the case of infinite sequences of binary variables, but more general formulations exist, such as that of Hewitt and Savage [43], that extend this result to arbitrarily distributed real-valued random variables.

This formulation provides important insight into the nature of exchangeability. Specifically, it reveals the surprising fact that given the parameter θ, the data $x_1, x_2, \ldots, x_n$ is conditionally independent and identically distributed (IID). In other words, the joint probability distribution $p(x_1, x_2, \ldots, x_n)$ of an exchangeable sequence can always be represented as a mixture of IID sequences of random variables with mixing distribution $p(\theta)$. The key thing to notice here is that the parameterization of θ is not restricted in complexity; in fact, the parameterization may need to be infinite-dimensional, providing motivation for the search for nonparametric Bayesian methods.

Thus, we have seen that exchangeable data can be viewed as being conditionally IID from a mixture, providing an analogue for IID data in the frequentist paradigm, and thus making efficient inference possible for this type of data. However, such mixtures may be arbitrarily complex, and may even require an infinite-dimensional parameterization to describe. Next, we will examine how to formalize and perform inference over such distributions.

2.3.2 The Chinese Restaurant Process and the Dirichlet Process

The Chinese restaurant process (CRP) [3] is a discrete-time stochastic process that produces exchangeable data and is often used to illustrate a generative model for cluster data in Bayesian nonparametrics. Informally, it can be imagined as a Chinese restaurant that contains in infinite number of empty tables that each have infinite capacity; at each time step, a customer enters the restaurant and either joins a previously occupied table, or sits at a new empty table. The i^{th} customer will choose to sit at an empty table with probability $\frac{\alpha}{i-1+\alpha}$ (all empty tables are identical; this is a *total* probability for sitting at any of them), where α is a "concentration" parameter that determines the degree of dispersion of customers to different tables. The probability of the customer instead choosing a particular occupied table z is proportional to the number of people s_z already sitting at it, causing a clustering or "rich-get-richer" effect, and is defined as $\frac{s_z}{i-1+\alpha}$. Additionally, the first person to sit at each table chooses a unique dish for everyone at the table to share from an infinite menu, where the dish corresponds to a vector of parameters θ_i that parameterize some distribution $F(\cdot)$. In other words, in terms of a clustering problem, each customer is a data point, each table is a cluster, or mixture component in a mixture model, and the table's dish represents the parameters of that mixture component (for example, the mean and variance of a Gaussian).

The sequence of customer assignments generated by the CRP is not IID, but it is exchangeable, so according to de Finetti's theorem, there exists a representation of the sequence that is conditionally IID with respect to some parameter, which in this case is θ. The mixing distribution G over the various possible settings of θ_i is simply defined by the number of customers sitting at the each of the corresponding tables. Thus, since each θ_i is a parameterization of a distribution, or a mixture component, G can be viewed as the mixing distribution of a mixture model.

However, there are an infinite number of tables, so G must be an infinite dimensional categorical distribution (a special case of a multinomial distribution where the number of trials is equal to 1). To take a Bayesian perspective on inference in such a model, we must specify a prior on G. If G were a fixed-dimensional, finite categorical distribution, then the appropriate conjugate prior would be a Dirichlet distribution. In this case, the appropriate prior is an infinite dimensional extension of the Dirichlet distribution, the Dirichlet Process.

Let us consider this model from a generative Bayesian perspective. The generative model for observations generated from a CRP partitioning is called a Dirichlet Process Mixture Model (DPMM) and can be specified as follows. A Dirichlet process (DP), parameterized by a base distribution G_0 over a parameter space $\boldsymbol{\theta}$, and a concentration parameter α, is used as a prior over the distribution G of mixture components. For data points X, mixture component parameters θ, and a parameterized distribution $F(\cdot)$, the DPMM can be written as [66]:

$$\begin{aligned}
&G|\alpha, G_0 \sim DP(\alpha, G_0)\\
&\theta_i|G \sim G\\
&x_i|\theta_i \sim F(\theta_i).
\end{aligned}$$

A draw G from a Dirichlet process is discrete with probability 1, as shown in the example in Figure 2.4. This draw provides the set of *atoms*, or mixture components, to which data points are assigned according to the distribution G, corresponding to the assignment of customers to tables in the CRP. Finally, each data point x_i can be generated by drawing from the distribution parameterized by the assigned mixture component θ_i. In the CRP analogy, this means that a customer x_i walks in and sits at the table with dish θ_i (in the generative view, the tables/dishes can be seen as generating the customers). Note that G must integrate to 1. From this, it can be seen that the Dirichlet process can be interpreted as a distribution over probability distributions. Figure 2.5 shows the corresponding graphical model for the DPMM. This model will be used in Chapter 3 to cluster state visitation data to find an appropriate number of subgoals for a task.

To show the relationship between the Dirichlet process and the CRP, let us first examine the simpler case of a finite mixture, defining it in such a way that the assignment of data to components is more explicit. Define $c_i \in \{1 \dots k\}$ as the index

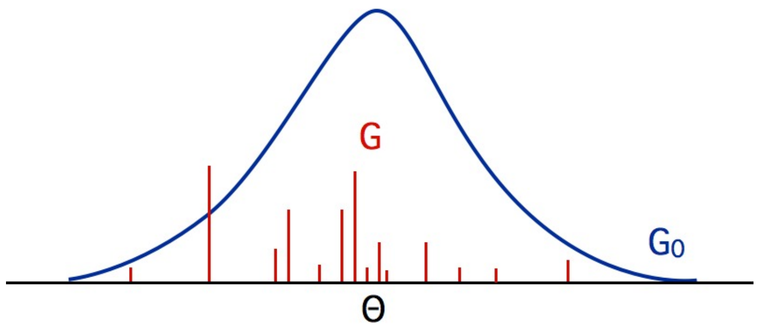


Figure 2.4. A discrete draw G from a Dirichlet process parameterized by G_0 over a parameter space $\boldsymbol{\theta}$.

of the mixture component assigned to observation x_i , where k is the number of mixture components, and π_j as the mixing weight on the j^{th} component. Thus, the generative model for this finite mixture is:

$$\begin{aligned}
&\theta_i | G_0 \sim G_0\\
&\boldsymbol{\pi} \sim \text{Dirichlet}(\alpha/k, \ldots, \alpha/k)\\
&c_i | \boldsymbol{\pi} \sim \text{Categorical}(\boldsymbol{\pi})\\
&x_i | \boldsymbol{\theta}, c_i \sim F(\theta_{c_i}).
\end{aligned}$$

Thus, the conditional probability of a mixture assignment[3] c_i, given all the other assignments $\mathbf{c}_{-i}$, can be found by integrating out the weights $\boldsymbol{\pi}$:

$$p(c_i = z \mid \mathbf{c}_{-i}, \alpha) = \int p(c_i = z \mid \boldsymbol{\pi}) p(\boldsymbol{\pi} \mid \mathbf{c}_{-i}, \alpha) d\boldsymbol{\pi}. \tag{2.6}$$

The first term in the integral, is simply equal to the z^{th} component of the weight vector:

[3]Note that due to the exchangeable nature of the data, we can always assume that the observation c_i in question is observed last given all the others, such that i=N, the total number of observations.

$$p(c_i = z \mid \boldsymbol{\pi}) = \pi_z. \tag{2.7}$$

The second term is the posterior probability of the weight vector, given all but the i^{th} mixture assignment, and can be written as:

$$p(\boldsymbol{\pi} \mid \mathbf{c}_{-i}, \alpha) \propto p(\mathbf{c}_{-i} \mid \boldsymbol{\pi}) p(\boldsymbol{\pi} \mid \alpha). \tag{2.8}$$

Since the first term has a categorical distribution and the second term is Dirichlet distributed, by conjugacy, the posterior $p(\boldsymbol{\pi} \mid \mathbf{c}_{-i}, \alpha)$ is also Dirichlet distributed. Define the normalization function for the Dirichlet distribution as:

$$\begin{aligned} \mathcal{Z}(\boldsymbol{\beta}) = & \int \prod_{j=1}^{k} \pi_j^{\beta_j - 1} d\pi & (2.9) \\ = & \frac{\prod_{j=1}^{k} \Gamma(\beta_j)}{\Gamma(\sum_{j=1}^{k} \beta_j)}, & (2.10) \end{aligned}$$

where $\Gamma(\cdot)$ is the standard Gamma function and β_j is a concentration parameters that can be conceptualized as a pseudocount of the number of times that the j^{th} event has previously been observed. Define the vector $\mathbf{s} = (s_1, \ldots, s_k)$, where s_j is the total number of assignment variables in $\mathbf{c}_{-i}$ that indicate mixture component j. Using this, the posterior probability of the weight vector can be written in terms of the previous number of counts, $\mathbf{s}$:

$$p(\boldsymbol{\pi} \mid \mathbf{c}_{-i}, \alpha) = \frac{1}{\mathcal{Z}(\mathbf{s} + \frac{\alpha}{k})} \prod_{j=1}^{k} \pi_j^{s_j + \frac{\alpha}{k} - 1}. \tag{2.11}$$

Finally, combining equations 2.7 and 2.11, and using the fact that $\Gamma(x+1) = x\Gamma(x)$, we can rewrite the posterior for the assignment variables (equation 2.6) as:

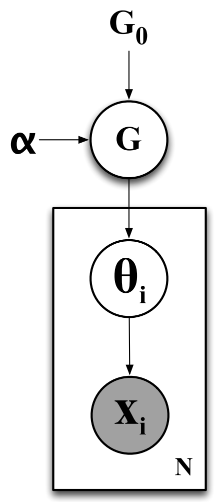


Figure 2.5. A Dirichlet Process Mixture Model (DPMM), where the rectangle is *plate notation*, denoting N copies of the outlined structure

$$
\begin{aligned}
&p(c_i = z \mid \mathbf{c}_{-i}, \alpha) && (2.12)\\
&\propto \frac{1}{\mathcal{Z}(\mathbf{s} + \frac{\alpha}{k})} \int \pi_z \prod_{j=1}^{k} \pi_j^{s_j + \frac{\alpha}{k} - 1} d\boldsymbol{\pi} && (2.13)\\
&= \frac{\mathcal{Z}(\mathbf{s} + \frac{\alpha}{k} + \mathbf{1}^{(z)})}{\mathcal{Z}(\mathbf{s} + \frac{\alpha}{k})} && (2.14)\\
&= \frac{\prod_{j=1}^{k} \Gamma\left(s_j + \frac{\alpha}{k} + \mathbf{1}_j^{(z)}\right)}{\prod_{j=1}^{k} \Gamma\left(s_j + \frac{\alpha}{k}\right)} \frac{\Gamma\left(\sum_{j=1}^{k} s_j + \frac{\alpha}{k}\right)}{\Gamma\left(\sum_{j=1}^{k} s_j + \frac{\alpha}{k} + \mathbf{1}_j^{(z)}\right)} && (2.15)\\
&= \frac{\left[\prod_{j\neq z} \Gamma\left(s_j + \frac{\alpha}{k}\right)\right] \Gamma\left(s_z + \frac{\alpha}{k} + 1\right)}{\prod_{j=1}^{k} \Gamma\left(s_j + \frac{\alpha}{k}\right)} \frac{\Gamma\left(\sum_{j=1}^{k} s_j + \frac{\alpha}{k}\right)}{\left(\sum_{j=1}^{k} s_j + \frac{\alpha}{k}\right) \Gamma\left(\sum_{j=1}^{k} s_j + \frac{\alpha}{k}\right)} && (2.16)\\
&= \frac{\left[\prod_{j=1}^{k} \Gamma\left(s_j + \frac{\alpha}{k}\right)\right] \left(s_z + \frac{\alpha}{k}\right)}{\prod_{j=1}^{k} \Gamma\left(s_j + \frac{\alpha}{k}\right)} \frac{1}{\sum_{j=1}^{k} s_i + \frac{\alpha}{k}} && (2.17)\\
&= \frac{s_z + \frac{\alpha}{k}}{N - 1 + \alpha}, && (2.18)
\end{aligned}
$$

where N is the total number of observations (including the current one whose mixture assignment is in question), and $\mathbf{1}^{(z)}$ is a vector of length k with a 1 in the z^{th} position, and zeros elsewhere.

Now, the behavior of this posterior probability can be examined as the number of mixture components k goes to infinity. Since only a finite number of samples, $N-1$, have been observed so far, then only a finite number of the counts $s_1, \ldots, s_\infty$ are non-zero. This divides the mixture components into two sets: a set $\mathcal{Q}$ that contains components with non-zero counts and a set $\hat{\mathcal{Q}}$ that contains components with zero counts. First, consider the probability that a new observation gets assigned to one particular mixture component z that already has a non-zero count s_z:

$$\begin{aligned} p(c_i = z \in \mathcal{Q} \mid \mathbf{c}_{-i}, \alpha) = & \lim_{k \to \infty} \left(\frac{s_z + \frac{\alpha}{k}}{N - 1 + \alpha} \right) & (2.19) \\ = & \frac{s_z}{N - 1 + \alpha}. & (2.20) \end{aligned}$$

Next, consider the total probability that a new observation gets assigned to any component that does not already have an associated observation (i.e. all components z such that the corresponding count s_z is equal to zero):

$$\begin{aligned} p(c_i \in \hat{\mathcal{Q}} \mid \mathbf{c}_{-i}, \alpha) = & \lim_{k \to \infty} \left(\sum_{z \in \hat{\mathcal{Q}}} \frac{s_z + \frac{\alpha}{k}}{N - 1 + \alpha} \right) & (2.21) \\ = & \lim_{k \to \infty} \left(\sum_{z \in \hat{\mathcal{Q}}} \frac{\frac{\alpha}{k}}{N - 1 + \alpha} \right) & (2.22) \\ = & \frac{\alpha}{N - 1 + \alpha} \lim_{k \to \infty} \left(\sum_{z \in \hat{\mathcal{Q}}} \frac{\frac{1}{k}}{N - 1 + \alpha} \right) & (2.23) \\ = & \frac{\alpha}{N - 1 + \alpha} \lim_{k \to \infty} \left(\frac{|\hat{\mathcal{Q}}|}{k(N - 1 + \alpha)} \right) & (2.24) \\ = & \frac{\alpha}{N - 1 + \alpha}, & (2.25) \end{aligned}$$

since $|\hat{\mathcal{Q}}| \to \infty$ as $k \to \infty$.

It can be seen that these probabilities are identical to those in the CRP for a customer joining an occupied table and an unoccupied table, respectively, showing the correspondence between the CRP and the Dirichlet process. Thus, the CRP is an illustrative description of a Bayesian prior over an infinite dimensional categorical distribution—in other words, the Dirichlet process. Equations 2.20 and 2.25 also illustrate that most of the data will have a tendency to cluster into a small number of components, since the likelihood of a new component forming becomes very small as N gets large. In fact, it can be shown that the number of non-zero components increases roughly logarithmically with respect to N [65].

Given a data set, inference can be performed on a DPMM to find an appropriate number of mixture components and their associated parameters that best explain the data without overfitting (of course, subject to the clustering strength assumptions made by setting the concentration parameter or its hyperparameters), in a fully Bayesian manner. This sidesteps the difficult problem of model selection and provides a principled framework for representing distributions of arbitrary complexity with mixtures of simpler distributions, such as Gaussians. This technique can be used for tasks such as unsupervised clustering of data and density estimation of complex distributions from samples. For a more complete discussion of how to perform inference in such a model, or how the full DPMM can be derived as the limit of finite mixture models, see [86].

2.3.3 The Chinese Restaurant Franchise and the Hierarchical Dirichlet Process

Now that the connection between the CRP and the Dirichlet process has been established, other similar metaphors can be explored that describe new classes of Bayesian nonparametric priors and their properties. One possible shortcoming of the standard Dirichlet process mixture model is that all component parameters θ_i are drawn from the same distribution G—in other words, that all data is drawn from the same underlying mixture distribution. However, in many data sets, data may be come from several distinct, but related, groups or distributions. By explicitly modeling these groups, the underlying distribution of the data can often be estimated more accurately and efficiently.

For example, imagine the distribution over words from a physics textbook compared to that of a chemistry textbook. Each component in a mixture model describing either book could represent a topic that generates words; since both books are science books, many such topics would appear in both books. Thus, to model the joint distribution of words across both books, it would be desirable to have a model that allowed some parameters, or *atoms*, to be shared across books, while still being able to model each book as having unique topics and distributions of topics. The Hierarchical Dirichlet Process, and its corresponding metaphor, the Chinese restaurant franchise (CRF) [98], allow such sharing.

The CRF can be described similarly to the CRP, but can share mixture components amongst groups (of data) that may have different, but related, characteristics. The primary difference is that the CRF assigns each group (or set of customers) to a separate restaurant. Each restaurant still has an infinite number of tables, but instead of each table's dish being globally unique, a dish can be chosen at additional tables at other restaurants in the franchise. Again, the first person to sit at each table chooses that table's dish from the menu, but in the CRF, the probability of choosing a particular dish is proportional to the number of tables that have already chosen that dish franchise-wide. Thus, dishes (i.e. mixture components such as topics) can be shared across restaurants, but each restaurant can have a unique distribution of dishes.

Just as the CRP describes a Dirichlet process prior, the CRF corresponds to a hierarchical Dirichlet process prior (HDP). The generative model for data sets produced by an HDP mixture model can be written as [98]:

$$
\begin{aligned}
&G_0|\gamma, H \sim DP(\gamma, H)\\
&G_j|\alpha, G_0 \sim DP(\alpha, G_0)\\
&\theta_{ji}|G_j \sim G_j\\
&x_{ji}|\theta_{ji} \sim F(\theta_{ji}),
\end{aligned}
$$

where H is a base distribution, γ is a concentration parameter, j is the index of a data group (i.e. an index that corresponds to all data generated from the atoms of a particular G_j), and both G_0 and G_j are discrete distributions. Here, the Dirichlet process that serves as a prior over G_0 defines a prior over the entire parameter space, whereas G_0 and G_j represent the franchise-wide and restaurant-specific menus, respectively. This model is said to be hierarchical since there are two levels (i.e. draws that take place in the generative model) at which atoms are selected that are then later used at lower levels; the atoms from G_0 can be shared amongst the various G_j, since G_0 is used as a base distribution to parameterize a second Dirichlet process that acts as a prior over G_j, encouraging sharing amongst the groups. Returning to our earlier example, it can be seen that an HDP mixture can be used to jointly model the word distribution from several documents that may share some topics, while also retaining unique topics and distributions over topics as well.

HDP priors also can be useful for modeling time-series data by acting as a prior over the transition matrix in a Hidden Markov model, forming an HDP-HMM [9]. In this case, the groups being modeled are the rows of the transition matrix; in other words, state-specific transition distributions. The HDP prior allows each transition distribution to be unique, while sharing global characteristics such as the overall popularity of particular states. Since the nonparametric prior allows this transition matrix to be infinite, an appropriate number of states (and their corresponding emission distributions) can be found that best explain the data without overfitting.

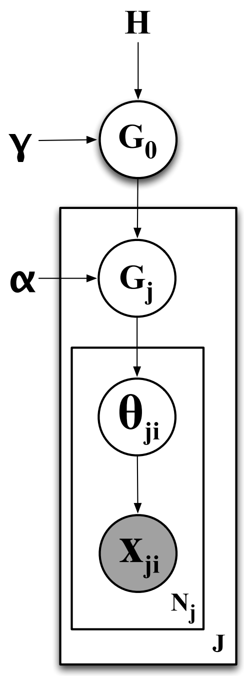


Figure 2.6. A Hierarchical Dirichlet Process Mixture Model (HDPMM)

2.3.4 The Indian Buffet Process and the Beta Process

The HDP-HMM is an effective model for analyzing a single time series. However, it is often desirable to be able to jointly analyze multiple time series sequences, each of which may have unique transition dynamics—for instance, demonstrations of several different robotic tasks. Furthermore, each sequence may only exhibit some subset of the larger set of states that are observed across all the sequences. It is relatively straightforward to extend the HDP-HMM so that it jointly models the transition and emission parameters of all the sequences. However, such a model assumes that all the sequences exhibit the same set of states and transition between them in an identical manner, precluding the desired flexibility. Instead, a more flexible *featural* model is required, in which each sequence can exhibit some subset of a larger library of states and transition between them in a unique manner.

The culinary metaphor that describes a statistical model with the properties that we desire is the Indian Buffet Process (IBP) [37]. In the IBP, the first customer enters the restaurant and select Poisson(α) dishes from an infinite buffet. After that, the i^{th} customer can select multiple dishes from the buffet; each existing dish z is selected with probability $\frac{s_z}{i}$, where s_z is the number of times dish z has previously been selected. The customer then also selects Poisson(α/i) new dishes. The IBP describes a nonparametric Bayesian prior known as the Beta process.

Recall that a draw from a Dirichlet process is a probability distribution—a draw from a Dirichlet process results in an infinite number of weights on point masses that sum to 1. A draw from a Beta process is also an infinite collection of point masses, but the weight of each point is drawn from a Beta distribution parameterized by the value of the base distribution at that point. Thus, the weights from a Beta process draw can each be interpreted as a $[0, 1]$ probability and do not sum to one. This draw can be seen as an infinite vector of probabilities corresponding to the chance of "heads" on an infinite set of unfair coins. The featural property of Beta processes stems from this view—in the generative view, when generating any given piece of data, a particular feature manifests with probability proportional to its weight in the draw from the Beta process.

A Beta process prior can be used over the transition matrices of hidden Markov models for multiple time series sequences, much like the HDP, but elicits a featural representation, in which hidden states can be shared across sequences. One such model is the Beta Process Autoregressive Hidden Markov Model (BP-AR-HMM) [33], shown in Figure 2.7, in which hidden states correspond to dynamic modes defined by linear dynamical systems. The BP-AR-HMM is able to jointly model a library of dynamical modes over many time series sequences, while allowing each time series to exhibit some subset of those modes and switch between them in a unique manner. Thus, a potentially infinite library of modes can be constructed in a fully Bayesian way, in which modes are flexibly shared between time series, and an appropriate number of modes is inferred directly from the data, without the need for model selection. In other words, each time series corresponds to a customer in the IBP and each dynamical mode corresponds to a dish.

The generative model for the BP-AR-HMM can be summarized as follows [33]:

$$
\begin{aligned}
&B|B_0 \sim \mathrm{BP}(1, B_0)\\
&X_i|B \sim \mathrm{BeP}(B)\\
&\pi_j^{(i)}|\boldsymbol{f_i}, \gamma, \kappa \sim \mathrm{Dir}([\gamma, ..., \gamma+\kappa, \gamma, ...] \otimes \boldsymbol{f_i})\\
&z_t^{(i)} \sim \pi^{(i)}_{z^{(i)}_{t-1}}\\
&\mathbf{y}_t^{(i)} = \sum_{j=1}^{r} A_{j, z_t^{(i)}} \mathbf{y}_{t-j}^{(i)} + \mathbf{e}_t^{(i)}(z_t^{(i)})
\end{aligned}
$$

First, a draw B from a Beta Process (BP) provides a set of global weights for the potentially infinite number of states. Then, for each time series, an X_i is drawn from a Bernoulli Process (BeP) parameterized by B. Each X_i can be used to construct a binary vector $\boldsymbol{f_i}$ indicating which of the global features, or states, are present in the i^{th} time series. Thus, B encourages sharing of features amongst multiple time series, while the X_i leave room for variability. Next, given the features that are present in each time series, for all states j, the transition probability vector $\pi_j^{(i)}$ is drawn from a Dirichlet distribution with self transition bias κ. A state $z_t^{(i)}$ is then drawn for each time step t from the transition distribution of the state at the previous time step. Finally, given the state at each time step and the *order* of the model, r, the observation is computed as a sum of state-dependent linear transformations of the previous r observations, plus mode-dependent noise. This model will be used in Chapters 4 and 5 to discover an appropriate number of dynamical systems that describe, and can be shared across, multiple observed robot demonstration trajectories.

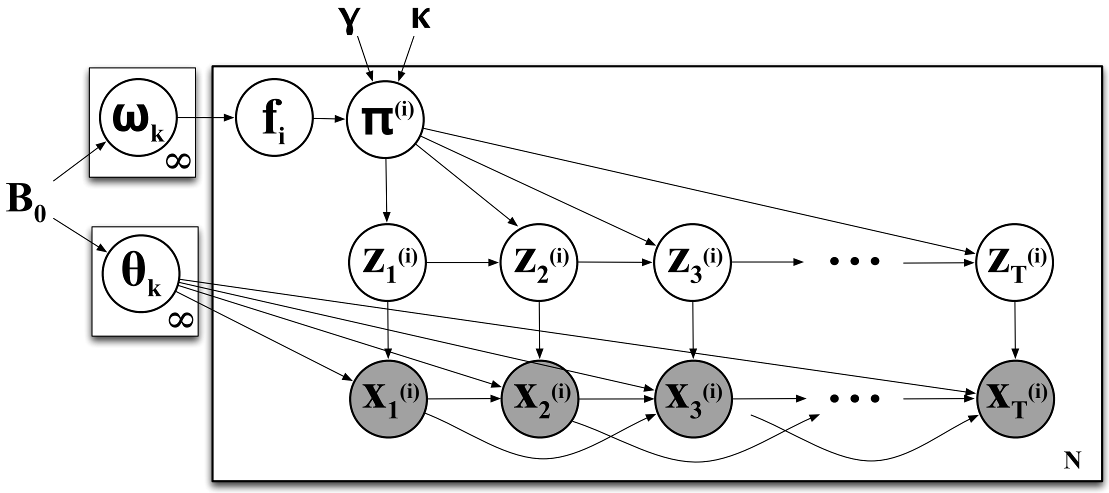


Figure 2.7. The Beta Process Autoregressive Hidden Markov Model (BP-AR-HMM). Here, the Beta Process draw B has been separated into masses $\boldsymbol{\omega}$ and parameters $\boldsymbol{\theta}$. Each parameter θ_k consists of r transition matrices $\mathbf{A}$ and a covariance term $\mathbf{e}$.

2.3.5 Nonparametric Bayesian Inference

Several nonparametric Bayesian methods have been discussed that can model various types of data of unknown complexity and sidestep the problem of model selection in a principled Bayesian manner. Thus far, these models have been explored from a generative point of view, with little discussion of how inference can be performed. In general, efficient Bayesian inference is model-specific and tailored to the unique statistical properties of the model. However, there are several core techniques and principles that are generally useful in performing interference in nonparametric Bayesian models, which we will now introduce.

First, how is it possible to perform inference on a model that has an infinite number of parameters? While it is not possible to optimize an infinite number of parameters, for problems like clustering with the DPMM, where only the assignments of data points to mixture components matter, it is possible to integrate the parameters out and look for high-likelihood configurations of the auxiliary variables that assign each observed data point to a mixture component. Other models like the BP-AR-HMM use incremental techniques like birth and death proposals [33] so that there are only a finite number of parameters to work with at any given time; the number of parameters can incrementally grow and shrink unboundedly, but is always practically limited by the size of the data.

Inferring the MAP distribution of the hidden variables in nonparametric models is often difficult, especially in the non-conjugate case. However, in applications like clustering, only unbiased samples from the posterior are required, rather than the full distribution. Leveraging this, Markov chain Monte Carlo (MCMC) sampling methods can be used to draw samples from the posterior when inferring the full posterior is impossible or intractable due to problems like high-dimensionality, difficulty of integration, and non-conjugacy.

Define z_i as the i^{th} variable in a model and $\mathbf{z}_{-i}$ as the set of all other variables in $\mathbf{z}$ except z_i. In cases where the conditionals $p(z_i|\mathbf{z}_{-i})$ cannot be written as a known distribution that can easily be sampled from, Metropolis-Hastings sampling [60, 42] can be used to obtain samples of the joint distribution $p(\mathbf{z})$. Algorithm 1 describes Metropolis-Hastings sampling, given a starting configuration of the variables $\mathbf{z}^{(0)}$, a *proposal distribution* $q(\mathbf{z}|\mathbf{z}')$ that is easy to sample from, and an unnormalized distribution $\widetilde{p}(\mathbf{z}) = \frac{1}{\mathcal{Z}}p(\mathbf{z})$, where $\mathcal{Z}$ may be unknown. It is assumed that while $p(\mathbf{z})$ cannot easily be sampled from, $\widetilde{p}(\mathbf{z})$ can be easily evaluated at a single point.

Under mild conditions, the Metropolis-Hastings algorithm creates a Markov chain whose stationary distribution approximates $p(\mathbf{z})$. It does this through a careful choice of the acceptance probability function that determines whether a step in the Markov chain is accepted or rejected. This function is designed so that the distribution being sampled at each time step is invariant and equal to the correct distribution $p(\mathbf{z})$. For more details on the derivation, see [12].

The Metropolis-Hastings algorithm has several drawbacks, most notably that it requires a "burn-in period"—a number of early samples that are thrown out, because of the bias introduced by the starting configuration. Additionally, successive samples are correlated, so if independent samples are desired, some number of samples must be ignored between each independent sample. An appropriate number can often be found by looking at the autocorrelation of samples.

Algorithm 1 Metropolis-Hastings Sampling

```
Given: Distributions $\widetilde{p}(\mathbf{z}), q(\mathbf{z}|\mathbf{z}')$
Input: Starting configuration $\mathbf{z}^{(0)}$

1. Initialize $\mathbf{z}^{(0)}$
2. For $\tau = 0, \ldots, T$:
    (a) Sample $\mathbf{z}^* \sim q(\mathbf{z}|\mathbf{z}^{(\tau)})$
    (b) Calculate acceptance probability:
        $A(\mathbf{z}^*, \mathbf{z}^{(\tau)}) = \min\left(1, \frac{\widetilde{p}(\mathbf{z}^*)q(\mathbf{z}^{(\tau)}|\mathbf{z}^*)}{\widetilde{p}(\mathbf{z}^{(\tau)})q(\mathbf{z}|\mathbf{z}^{(\tau)})}\right)$
    (c) Sample $u \sim \text{uniform}(0, 1)$
    (d) Assign $\mathbf{z}^{(\tau+1)}$:
        if $A(\mathbf{z}^*, \mathbf{z}^{(\tau)}) > u$ then
            $\mathbf{z}^{(\tau+1)} = \mathbf{z}^*$
        else
            $\mathbf{z}^{(\tau+1)} = \mathbf{z}^{(\tau)}$
        end if
3. Return $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(T+1)}$
```

When the conditional distributions $p(z_i|\mathbf{z}_{-i})$ can be written in a standard form for which the CDF can easily be calculated, a special case of Metropolis-Hastings sampling called Gibbs sampling [34] can be used. Algorithm 2 outlines the Gibbs sampling procedure. Gibbs sampling uses the conditionals $p(z_i|\mathbf{z}_{-i})$ as proposal distributions, cycling through and sampling from them one at a time. Instead of calculating an acceptance probability, it can be shown that under this choice of proposal distribution, the new draw should always be accepted. Thus, when the conditionals are available in a standard form, Gibbs sampling can show substantial efficiency gains over more general Metropolis-Hastings sampling.

Algorithm 2 Gibbs Sampling

```
Given: Conditionals $p(z_1|\mathbf{z}_{-1}), \ldots, p(z_M|\mathbf{z}_{-M})$
Input: Starting configuration $\mathbf{z}^{(0)}$

  1. Initialize $\mathbf{z}^{(0)}$

  2. For $\tau = 0, \ldots, T$:
       Sample $z_1^{(\tau+1)} \sim p(z_1|z_{-1}^{(\tau)})$
       $\vdots$
       Sample $z_M^{(\tau+1)} \sim p(z_M|z_{-M}^{(\tau)})$

  3. Return $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(T+1)}$
```

The full details of inference for the models described earlier are outside the scope of this document, but there exist many good references on this topic [30, 65, 66, 86].

CHAPTER 3
PORTABLE SUBGOAL DISCOVERY

3.1 Introduction

One of the simplest types of demonstrations are those in which the user specifies domain-specific events that are task-critical, but does not need to demonstrate how these events can be brought about. This requires the user to have some descriptive knowledge about important features of the task but does not require them to be skilled at the task itself. This method of demonstration places little onus on the user and can be useful for tasks that are difficult for humans, as well as tasks in which the number of different approaches are difficult to capture programmatically.

For example, consider the potentially useful subgoal of capturing the queen in chess. A beginner at chess may know that this subgoal is useful, but not know how to effectively accomplish it in many situations; an expert may know how to capture the queen, but may have trouble explicitly characterizing all the possible paths to this goal. Thus, it can be more efficient to simply specify the critical events, allow the agent to discover an appropriate number of skills to bring about these events in various situations, and use *reinforcement learning* to allow the agent to automatically learn and improve skill policies from experience.

Reinforcement learning (RL) is often used to solve single tasks for which it is tractable to learn a good policy with minimal initial knowledge. However, many real-world problems cannot be solved in this fashion, motivating recent research on transfer and hierarchical RL methods that allow knowledge to be generalized to new problems and encapsulated in modular skills. Although skills have been shown to improve agent learning performance [7], representational power [55], and adaptation to non-stationarity [25], most current methods lack the ability to automatically discover skills that are transferable to related state spaces and novel tasks, especially in continuous domains.

Skill discovery algorithms in reinforcement learning typically identify single states or regions in state space that correspond to task-specific subgoals. However, such methods do not directly address the question of how many distinct skills are appropriate for solving the tasks that the agent faces. This can be highly inefficient when many identified subgoals correspond to the same underlying skill, but are all used individually as skill goals. For example, opening a door ought to be the same skill whether an agent is one inch or two inches away from the door, or whether the door is red or blue; making each possible configuration a separate skill would be unwise. Furthermore, skills created in this manner are often only transferable to tasks that share identical state spaces, since corresponding subgoals across tasks are not merged into a single skill goal.

In this chapter, we show that these problems can be overcome by collecting subgoal data from a series of tasks and clustering it in an *agent-space* [54], a shared feature space across multiple tasks. The resulting clusters generalize subgoals within and across tasks and can be used as templates for portable skill termination conditions. Clustering also allows the creation of skill termination conditions in a data-driven way that makes minimal assumptions and can be tailored to the domain through a careful choice of clustering algorithm. Additionally, this framework extends the utility of single-state subgoal discovery algorithms to continuous domains, in which the agent may never see the same state twice. We argue that clustering based on a Dirichlet process mixture model is appropriate in the general case when little is known about the nature or number of skills needed in a domain. Experiments in a continuous domain demonstrate the utility of this approach and illustrate how it may be useful even when traditional subgoal discovery methods are infeasible.

3.2 Background

3.2.1 Reinforcement learning

Reinforcement learning (RL) [96] is a well-studied methodology that allows agents to learn to maximize a reward signal through experience in an environment. The RL paradigm usually models a problem faced by the agent as a Markov decision process (MDP), expressed as $M = \langle S, A, P, R \rangle$, where S is the set of environment states the agent can observe, A is the set of actions that the agent can execute, $P(s, a, s')$ is the probability that the environment transitions to $s' \in S$ when action $a \in A$ is taken in state $s \in S$, and $R(s, a, s')$ is the expected scalar reward given to the agent when the environment transitions to state s' from s after the agent takes action a.

Given the reward function, the agent's objective is to learn the optimal policy π^* that maximizs a measure of the cumulative reward it receives. Commonly, this measure is the *discounted cumulative reward*, defined as $\sum_{i=0}^{\infty} \gamma^i r_{t+i+1}$, where r_t is the reward received at time t and $0 \leq \gamma < 1$ is a discount factor that specifies to what degree the agent prefers immediate rewards to future ones.

3.2.2 Options

The options framework [95] models skills as temporally extended actions that can be invoked like primitive actions. An option o consists of an *option policy* $\pi_o : S \times A \rightarrow [0, 1]$, giving the probability of taking action a in state s, an *initiation set* $\mathcal{I}_o \subseteq S$, giving the set of states from which the option can be invoked, and a *termination condition* $\beta_o : S \rightarrow [0, 1]$, giving the probability that option execution will terminate upon reaching state s. In this chapter, termination conditions are binary, so that we can define a *termination set* of states, $\mathcal{T}_o \subseteq S$, in which option execution always terminates.

3.2.3 Agent-spaces

To facilitate option transfer across multiple tasks, Konidaris and Barto [54] propose separating problems into two representations. The first is a *problem-space* representation which is Markov for the current task being faced by the agent, but may change across tasks; this is the typical formulation of a problem in RL. The second is an *agent-space* representation, which is identical across all tasks to be faced by the agent, but may not be Markov for any particular task. An agent-space is often a set of agent-centric features, like a robot's sensor readings, that are present and retain semantics across tasks. The decision about which features to include in the agent-space typically requires some knowledge of the domain to determine which features are likely to have an invariant interpretation across problem instances. If the agent represents its top-level policy in a task-specific problem-space but represents its options in an agent-space, the task at hand will always be Markov while allowing the options to transfer between tasks.

Agent-spaces enable the transfer of an option's policy between tasks, but are based on the assumption that this policy was learned under an option termination set that is portable; the termination set must accurately reflect how the goal of the skill varies across tasks. Previous work using agent-spaces has produced portable option policies when the termination sets were hand-coded; our contribution is the automatic discovery of portable termination sets, so that such skills can be acquired autonomously.

3.2.4 Expectation-maximization

Expectation-maximization (E-M) [24] is a general method for finding maximum likelihood parameters for models with latent variables. E-M can be used to estimate the means $\mu_1 ... \mu_K$, covariances $\Sigma_1 ... \Sigma_K$, and mixing coefficients $\pi_1 ... \pi_K$ of a Gaussian mixture model [12], where we specify *a priori* a number of Gaussians, K, and are given data points $x_1 ... x_N$. A solution is iteratively computed through an expectation step defined by:

$$\gamma_{nk} = \frac{\pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)}{\sum\limits_{j=1}^{K} \pi_j \mathcal{N}(x_n | \mu_k, \Sigma_k)},$$

and a maximization step defined by:

$$\begin{aligned} N_k &= \sum_{n=1}^{N} \gamma_{nk} \\ \mu_k &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} x_n \\ \Sigma_k &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} (x_n - \mu_k)(x_n - \mu_k)^T \\ \pi_k &= \frac{N_k}{N}. \end{aligned}$$

These steps are iterated until convergence of the log likelihood:

$$\ln p(X | \mu, \Sigma, \pi) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k) \right)$$

After convergence, each data point x_n can be given a hard assignment to the cluster corresponding to the component with the maximum γ_{nk} for all k. This algorithm is subject to local minima, so it may be beneficial to run it multiple times and to use the resulting clustering with the highest log likelihood—in our experiments, we use 10 repetitions. Also, due to numerical precision issues, some of the quantities above must be approximated or rescaled when actually implementing the algorithm.

3.2.5 The Infinite Gaussian Mixture Model

Many popular clustering algorithms require the number of data clusters to be known *a priori* or use heuristics to choose an approximate number. By contrast, Dirichlet process mixture models (DPMMs) provide a non-parametric Bayesian framework to describe distributions over mixture models with an infinite number of mixture components. One type of DPMM can be implemented as an infinite Gaussian mixture model (IGMM) in which all parameters are inferred from the data [86]. The generative model for the IGMM can be written as:

$$\begin{aligned}
&G|\alpha, G_0 \sim DP(\alpha, G_0) \\
&\theta_i|G \sim G \\
&x_i|\theta_i \sim \mathcal{N}(\theta_i^{\mu}, \theta_i^{\Sigma}),
\end{aligned}$$

where parameters θ_i consist of a mean θ_i^{μ} and covariance θ_i^{Σ} of a Gaussian.

Gibbs sampling is used to generate samples from the posterior distribution of the IGMM and adaptive rejection sampling [36] is used for the probabilities which are not in a standard form. After a "burn-in" period, unbiased samples from the posterior distribution of the IGMM can be drawn from the Gibbs sampler. A hard clustering can be found by drawing many such samples and using the sample with the highest joint likelihood of the class indicator variables. We use a modified IGMM implementation written by M. Mandel.[1]

3.3 Latent Skill Discovery

To aid thinking about our algorithm, subgoals can be viewed as samples from the termination sets of *latent options* that are implicitly defined by the distribution of

[1]Source code can be found at `http://mr-pc.org/work/`

tasks, the chosen subgoal discovery algorithm, and the agent definition. Specifically, we define the latent options as those whose termination sets contain all of the sampled subgoal data and that maximize the expected discounted cumulative reward when used by a particular agent on a distribution of tasks (assuming optimal option policies given the termination sets). When many such maximizing sets exist, we assume that the latent options are one particular set from amongst these choices; for our discussion, the particular choice does not matter, but it is important to have a single set.

Therefore, our goal is to recover the termination sets of the latent options from the sampled subgoal data; these can be used to construct a library of options that approximate the latent options and have the following desirable properties:

- Recall: The termination sets of the library options should contain a maximal portion of the termination sets of the latent options.
- Precision: The termination sets of the library options should contain minimal regions that are not in the termination sets of the latent options.
- Separability: The termination set of each library option should be entirely contained within the termination set of some single latent option.
- Minimality: A minimal number of options should be defined, while still meeting the above criteria. Ideally, this will be equal to the number of latent options.

Most of these properties are straightforward, but the importance of separability should be emphasized. Imagine an agent that faces a distribution of tasks with several latent options that need to be sequenced in various ways for each task. If a clustering breaks each latent option termination set into two options (minimality is violated, but separability is preserved), some exploration inefficiency may be introduced, but each option will reliably terminate in a skill-appropriate state. However, if a clustering combines the termination sets of two latent options into that of a single library option, the library option becomes unreliable; when the functionality of a single latent option is needed, the combined option may exhibit behavior corresponding to either.

We cannot reason directly about latent options since we do not know what they are *a priori*, so we must estimate them with respect to the above constraints from sampled subgoal data alone. We assume that subgoal samples corresponding to the same latent option form a contiguous region on some manifold, which is reflected in the problem representation. If they do not, then our method cannot cluster and find skills; we view this as a failing of the representation and not of our methodology.

Under this assumption, clustering of sampled subgoals can be used to approximate latent option termination sets. We propose a method of converting clusters parameterized by Gaussians into termination sets that respect the recall and precision properties. Knowing the number of skills *a priori* or discovering the appropriate number of clusters from the data satisfies the minimality property. Separability is more complicated, but can be satisfied by any method that can handle overlapping clusters without merging them and that is not inherently biased toward a small number of skills. Methods like spectral clustering [69] that rely on point-wise distance metrics cannot easily handle cluster overlap and are unsuitable for this sort of task. In the general case where little is known about the number and nature of the latent options, IGMM-based clustering is an attractive choice, as it can model any number of clusters of arbitrary complexity; when clusters have a complex shape, an IGMM may over-segment the data, but this still produces separable options.

3.4 Algorithm

We present a general algorithm shown in Algorithm 3 to discover latent options when using any particular subgoal discovery method and clustering algorithm [73]. Note that some subgoal discovery methods discover state regions, rather than single states; in such cases, sampling techniques or a clustering algorithm such as NPClu

[41] that can handle non-point data must be used. We then describe a specific implementation of the general algorithm that is used in our experiments.

Algorithm 3 Portable Subgoal Discovery

Given: An agent $\mathcal{A}$, task distribution τ, subgoal discovery algorithm $\mathcal{D}$, and clustering algorithm $\mathcal{C}$

1. Compute a set of sample agent-space subgoals $X = \{x_1, x_2, ..., x_n\}$, where $X = \mathcal{D}(\mathcal{A}, \tau)$.
2. Cluster the subgoals X into clusters with parameters $\theta = \{\theta_1, \theta_2, ..., \theta_k\}$, where $\theta = \mathcal{C}(X)$. If the clustering method is parametric, then the elements of θ are cluster parameters, otherwise they are data point assignments to clusters.
3. Define option termination sets $\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_k$, where $\mathcal{T}_i = \mathcal{M}(\theta_i)$, and $\mathcal{M}$ is a mapping from elements of θ to termination set definitions.
4. Instantiate and train options $\mathcal{O}_1, \mathcal{O}_2, ..., \mathcal{O}_k$ using $\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_k$ as termination sets.

3.4.1 Experimental implementation

We now present an example implementation of the general algorithm that is used in our experiments. As to not confound error from our clustering method with error introduced by a subgoal discovery algorithm, we use a hand-coded binary salience function; the main contribution of this work is the clustering strategy that enables generalization and transfer, so we are not concerned with the details of any particular subgoal discovery algorithm. This also demonstrates the possible utility of our approach, even when automatic subgoal discovery is inappropriate or infeasible. More details on this are presented in the following sections.

First, a distribution of tasks and an RL agent are defined. We allow the agent to solve tasks drawn from this distribution while collecting subgoal state samples every time the salience function is triggered. This continues until 10,000 subgoal state samples are collected. These points are then clustered using one of two different clustering methods. Gaussian expectation-maximization (E-M), for which we must provide the number of clusters *a priori*, provides an approximate upper bound on the performance of any clustering method based on a Gaussian mixture model. We compare this to IGMM-based clustering that must discover the number of clusters automatically. E-M is used as a baseline metric to separate error caused by not knowing the number of clusters *a priori* from error caused by using a Gaussian mixture model. Since E-M can get stuck in local minima, we run it 10 times and choose the clustering with the highest log-likelihood. For the IGMM-based clustering, we let the Gibbs sampler burn-in for 10,000 samples and then collect an additional 10,000 samples, from which we choose the sample with the highest joint likelihood of the class indicator variables as defined by Rasmussen [86].

We now must define a mapping function $\mathcal{M}$ that maps our clusters to termination sets. Both of our clustering methods return a list of K sets of Gaussian means μ and covariances Σ. We would like to choose a ridge on each Gaussian to be the cluster's termination set boundary; thus, we use Mahalanobis distance from each cluster mean, where

$$D_i^{Mahalanobis}(x) = \sqrt{(x-\mu_i)^T \Sigma_i^{-1} (x-\mu_i)}\ ,$$

and the termination set $\mathcal{T}_i$ is defined as:

$$\mathcal{T}_i(x) = \begin{cases} 1 & : D_i^{Mahalanobis}(x) \leq \epsilon_i \\ 0 & : \text{otherwise,} \end{cases}$$

where ϵ_i is a threshold. An appropriate value for each ϵ_i is found automatically by calculating the maximum $D_i^{Mahalanobis}(x)$ of any of the subgoal state points x assigned to the i^{th} cluster. This makes each ϵ_i just large enough so that all the subgoal state data points assigned to the i^{th} cluster are within the ϵ_i-Mahalanobis distance of that cluster mean, satisfying both our recall and precision conditions. Note that some states can be contained in multiple termination sets.

Using these termination sets, we create options that are given to the agent for a 100 episode "gestation period", during which the agent can learn option policies using off-policy learning, but cannot invoke the options. After this period, the options can be invoked from any state.

3.5 Experiments

3.5.1 Light-Chain domain

We test the various implementations of our algorithm on a continuous domain similar to the Lightworld domain [54], designed to provide intuition about the capabilities of our skill discovery method. In our version, the Light-Chain domain, an agent is placed in a 10×10 room that contains a primary beacon, a secondary beacon, and a goal beacon placed in random locations. If the agent moves within 1 unit of the primary beacon, the beacon becomes "activated" for 30 time steps. Similarly, if the agent moves within 1 unit of the secondary beacon while the primary beacon is activated, it also becomes activated for 30 time steps. The goal of the task is for the agent to move within 1 unit of the goal beacon while the secondary beacon is activated, upon which it receives a reward of 100, ending the episode. In all other states, the agent receives a reward of -1. Additionally, each beacon emits a uniquely colored light—either red, green, or blue—that is selected randomly for each task. Figure 3.1 shows two instances of the Light-Chain domain with different beacon locations and light color assignments.

There are four actions available to the agent in every state: move north, south, east, or west. The actions are stochastic, moving the agent between 0.9 and 1.1 units (uniformly distributed) in the specified direction. In the case of an action that would move an agent through a wall, the agent simply moves up to the wall and stops. The problem-space for this domain is 4-dimensional: The x-position of the agent, the y-position of the agent, and two boolean variables denoting whether or not the primary and secondary beacons are activated, respectively. The agent-space is 6-dimensional and defined by RGB range sensors that the agent is equipped with. Three of the sensors describe the north/south distance of the agent from each of the three colored lights (0 if the agent is at the light, positive values for being north of it, and negative vales for being south of it). The other three sensors are identical, but measure east/west distance. Since the beacon color associations change with every task, a portable top-level policy cannot be learned in agent-space, but portable agent-space options can be learned that reliably direct the agent toward each of the lights.

The agent's salience function is defined as:

$$salient(t) = \begin{cases} 1 & : \text{At time } t\text{, a beacon became activated for the first time in this episode.} \\ 0 & : \text{otherwise.} \end{cases}$$

Our algorithm clusters subgoal state data to create option termination conditions that generalize properly within a task and across tasks. In the Light-Chain domain, there are three latent options—one corresponding to each light color. Generalization within a task requires each option to terminate in any state within a 1 unit radius of its corresponding light color. However, if the agent only sees one task, all such states will be within some small fixed range of the other two lights; a termination set built from such data would not transfer to another task, since the relative positions of the lights might change. Thus, generalization across tasks requires each option to terminate when it is close to the proper light, regardless of the observed positions of the other two lights. When provided with data from many tasks, our algorithm can discover these relationships between agent-space variables and use them to define portable options. These options can then be used in each task, although in a different order for each, based on that task's color associations with the beacons.

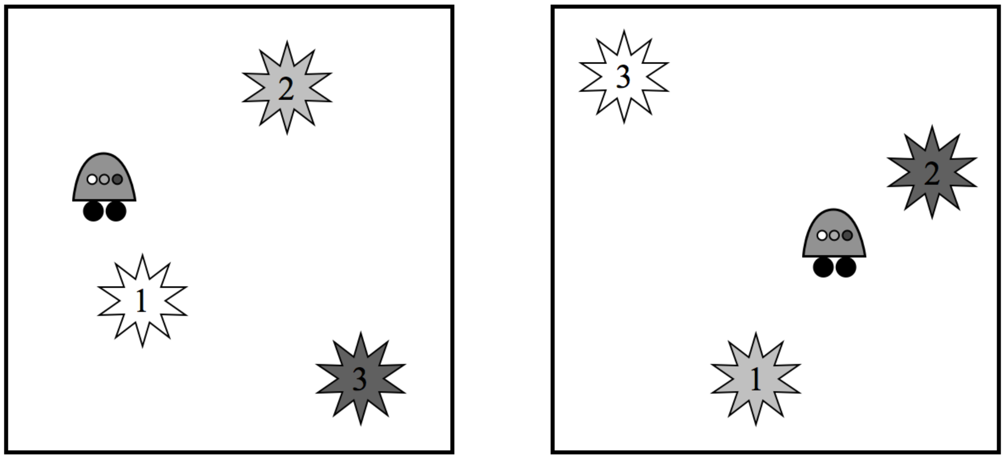


Figure 3.1. Two instances of the Light-Chain domain. The numbers 1–3 indicate the primary, secondary, and goal beacons respectively, while color signifies the light color each beacon emits. Notice that both beacon placement and color associations change between tasks.

Although we provide a broad subgoal (activate beacons) to the agent through the salience function, our algorithm does the work of discovering how many ways there are to accomplish these subgoals (three—one for each light color) and how to achieve each of these (get within 1 unit of that light). In each instance of the task, it is unknown which light color will correspond to each beacon. Therefore, it is not possible to define a skill that reliably guides the agent to a particular beacon (e.g. the primary beacon) and is portable across tasks. Instead, our algorithm discovers skills to navigate to particular lights, leading the agent to beacons by proxy. Note that this number of skills is independent of the number of beacons; if there were four possible colors of light, but only three beacons, four skills would be created so that the agent could perform well when presented with any three of the four colors in a given task. Similarly, such a setup can be used in other tasks where a broad subgoal is known, but the different means and number of ways of achieving it are unknown *a priori*.

3.5.2 Experimental structure

Two different agent types were used in our experiments: agents with and without options. The parameters for each agent type were optimized separately via a grid search. Top-level policies were learned using ϵ-greedy SARSA(λ) ($\alpha = 0.001$, $\gamma = 0.99$, $\lambda = 0.7$, $\epsilon = 0.1$ without options, $\alpha = 0.0005$, $\gamma = 0.99$, $\lambda = 0.9$, $\epsilon = 0.1$ with options) and the state-action value function was represented with a linear function approximator using the third order Fourier basis [53]. Option policies were learned off-policy (with an option reward of 1000 when in a terminating state), using Q(λ) ($\alpha = 0.000025$, $\gamma = 0.99$, $\lambda = 0.9$) and the fifth order decoupled Fourier basis.

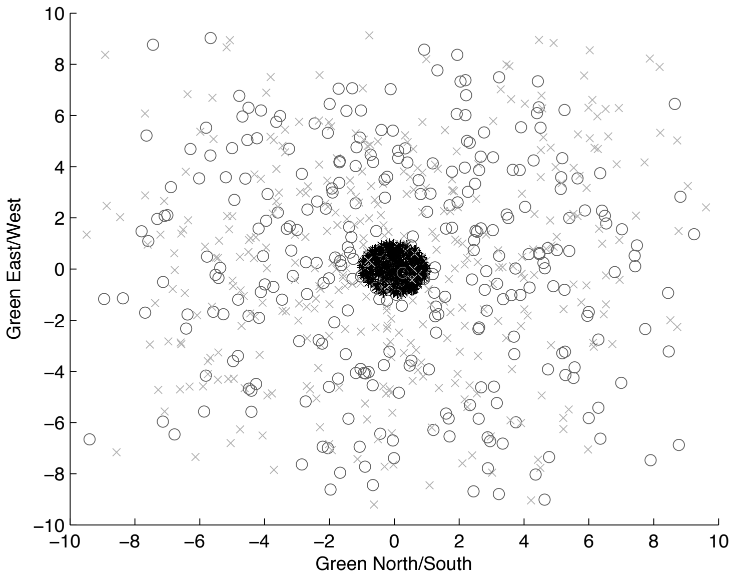
For the agents that discover options, we used the procedure outlined in the previous section to collect subgoal state samples and learn option policies. We compared these agents to an agent with perfect, hand-coded termination sets (each option terminated within 1 unit of a particular light) that followed the same learning procedure, but without the subgoal discovery step. After option policies were learned for 100 episodes, they were frozen and agent performance was measured for 10 episodes in each of 1,000 novel tasks, with a maximum episode length of 5,000 steps and a maximum option execution time of 50 steps. After each task, the top-level policy was reset, but the option policies were kept constant. We compared performance of the agents using options to that of an agent without options, tested under the same conditions. This entire experiment was repeated 10 times.

3.6 Results

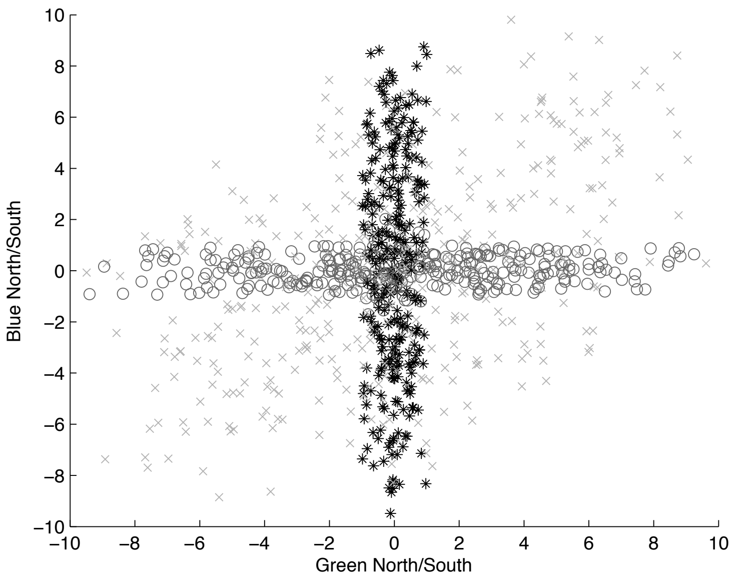
Figure 3.2 shows an IGMM-based clustering (only 1,000 points shown for readability), in which the original data points are projected onto 2 of the 6 agent-space dimensions at a time for visualization purposes, where cluster assignment is denoted with unique markers. It can be seen that three clusters (the intuitively optimal number) have been found. In 3.2(a), the data is projected onto the green north/south and green east/west dimensions. A central circular cluster is apparent, containing subgoals triggered by being near the green light. In 3.2(b), the north/south dimensions of two different light colors are compared. Here, there are two long clusters that each have a small variance with respect to one color and a large variance with respect to the other. These findings correspond to our intuitive notion of skills in this domain, in which an option should terminate when it is close to a particular light color, regardless of the positions of the other two lights. Note that these clusters actually overlap in 6 dimensions, not just in the projected view, since the activation radii of the beacons can occasionally overlap, depending on their placement.

Figure 3.3(a) compares the cumulative time it takes to solve 10 episodes for agents with no options, IGMM options, E-M options (with three clusters), and options with perfect, hand-coded termination sets. As expected, in all cases, options provide a significant learning advantage when facing a novel task. The agent using E-M options performs only slightly worse than the agent using perfect, hand-coded options, showing that clustering effectively discovers options in this domain and that very little error is introduced by using a Gaussian mixture model. Possibly more surprisingly, the agent using IGMM options performs equally as well as the agent using E-M options (making the lines difficult to distinguish in the graph), demonstrating that estimating the number of clusters automatically is feasible in this domain and introduces negligible error. In fact, the IGMM-based clustering finds three clusters in all 10 trials of the experiment.

Figure 3.3(b) shows the performance of agents using E-M options where the number of pre-specified clusters varies. As expected, the agent with three options (the intuitively optimal number of skills in this domain) performs the best, but the agents using five and six options still retain a significant advantage over an agent with no options. Most notably, when less than the optimal number of options are used, the agent actually performs worse than the baseline agent with no options. This confirms our

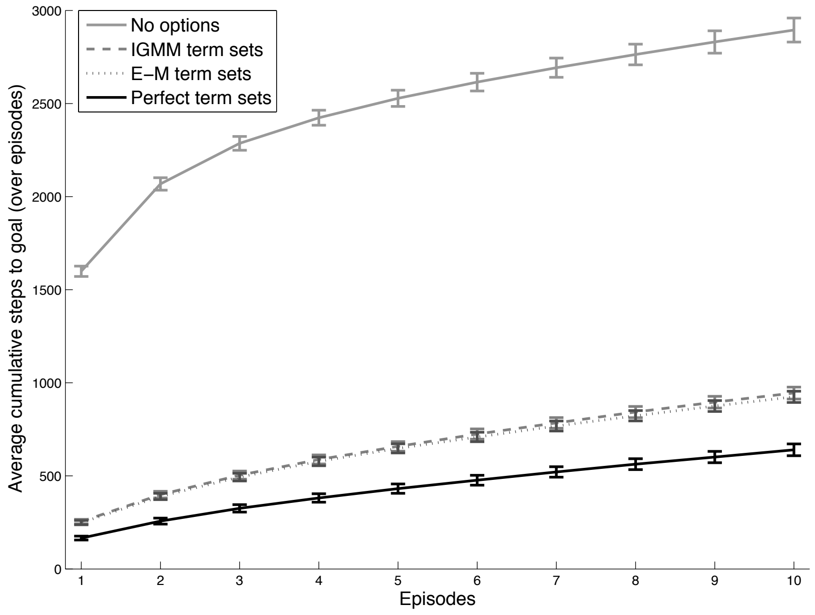


(a) Proj. onto Green-N/S and Green-E/W

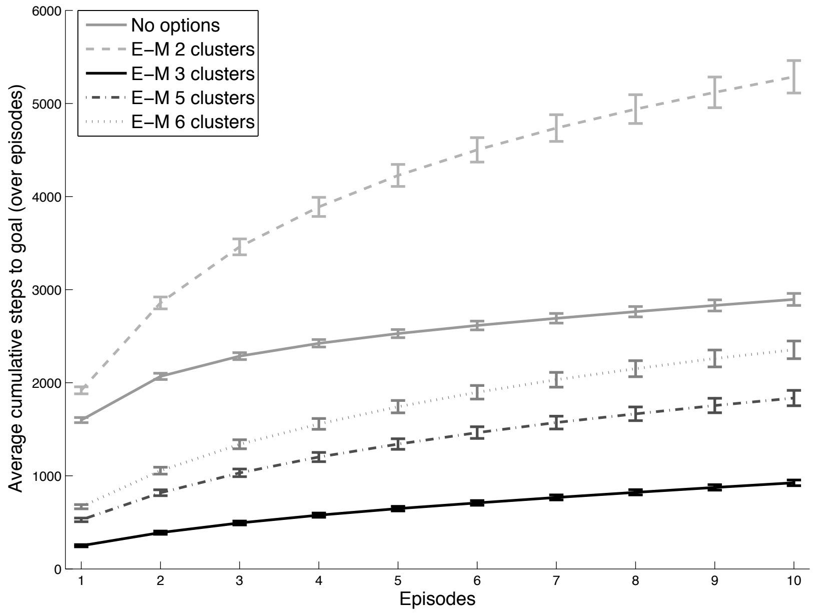


(b) Proj. onto Green-N/S and Blue-N/S

Figure 3.2. IGMM clusterings of 6-dimensional subgoal data projected onto 2 dimensions at a time for visualization. The color/shape of each point denotes cluster membership.



(a) Comparative performance of agents



(b) E-M with varying numbers of clusters

Figure 3.3. Agent performance in Light-Chain domain with 95% confidence intervals

intuition that option separability is more important than minimality. Thus, it seems that E-M may be effective if the designer can come up with a good approximation of the number of latent options, but it is critical to overestimate this number.

3.7 Discussion and Conclusions

We have demonstrated a general method for clustering agent-space subgoal data to form the termination sets of portable skills in the options framework. This method works in both discrete and continuous domains and can be used with any choice of subgoal discovery and clustering algorithms. Our analysis of the Light-Chain domain suggests that if the number of latent options is approximately known *a priori*, clustering algorithms like E-M can perform well. However, in the general case, IGMM-based clustering is able to discover an appropriate number of options automatically without sacrificing performance.

The collection and analysis of subgoal state samples can be computationally expensive, but this is a one-time cost. Our method is most relevant when a distribution of tasks is known ahead of time and we can spend computational time up front to improve agent performance on new tasks to be faced later, drawn from the same distribution. This can be beneficial when an agent will have to face a large number of related tasks, like in DRAM memory access scheduling [46], or for problems where fast learning and adaptation to non-stationarity is critical, such as automatic anesthesia administration [62].

In domains where traditional subgoal discovery algorithms fail or are too computationally expensive, it may be possible to define a salience function that specifies useful subgoals, while still allowing the clustering algorithm to decide how many skills are appropriate. Such a setup is advantageous when a broad subgoal is known *a priori*, but the various means and number of ways in which the subgoal might be accomplished are unknown, as in our Light-Chain experiment. This extends the possibility of skill discovery to a class of domains in which it may have previously been intractable.

We make an assumption that the subgoal state data is organized in a way such that an appropriate clustering algorithm can properly break it up into useful option termination sets; this is a function of both the problem representation and the clustering method. Therefore, future work may include using representation discovery techniques such as manifold learning to re-represent the data in a way that is easier to cluster. All clustering methods are biased in some way, but we have reason to believe that methods that preserve skill *separability* and favor a larger number of clusters are preferable. Given these criteria, clustering methods built on Dirichlet process mixture models are attractive, regardless of the shape, complexity, and overlap of the latent option termination sets.

This work also assumes the existence of an appropriate agent-space for the distribution of tasks. While this can be difficult to design by hand for some problems, previous work has proposed methods for automatically identifying agent-spaces embedded within problem-spaces [94]. Furthermore, defining an agent-space is trivial when an embodied agent is being used, as in robotic domains; the agent's sensors naturally define a feature space that is common to all tasks that the agent can face.

An agent with a library of appropriate portable options ought to be able to learn novel tasks faster than an agent without options. However, as this library grows, the number of available actions actually increases and agent performance may begin to decline. This counter-intuitive notion, commonly known as the *utility problem*, reveals a fundamental problem with using skills outside the context of hierarchies. For skill discovery to be useful in larger problems, future work will have to address basic questions about how to automatically construct appropriate skill hierarchies that allow the agent to explore in simpler, more abstract action spaces as it gains more skills and competency.

Finally, the most fundamental problem with this approach is revealed when trying to apply it in contexts where it is infeasible to collect a sufficient amount of subgoal data, as it is in many robotic domains. Unless a very good simulator is available, it is simply not possible to collect the amount of data necessary to properly characterize the skill termination conditions in the way that we have shown in our simulated experiment. This problem motivates the following chapters, in which we leverage the additional structure that can be found in full demonstrations of robotic tasks to discover useful, generalizable skills from a small number of examples.

CHAPTER 4

LEARNING FROM UNSTRUCTURED DEMONSTRATIONS

In the previous chapter, a nonparametric Bayesian algorithm was used to find an appropriate number of subgoals, or skill termination sets, allowing an agent to automatically learn skill policies using RL. However, this method proved to require large amounts of execution data, becoming intractable for problems involving physical robots. Fortunately, for tasks that can be demonstrated by an expert, a great deal of additional structure that can be leveraged to make skill discovery and learning more data efficient. In this chapter, we again use a nonparametric Bayesian model to discover skills, but within the context of learning from demonstration. This, combined with a coordinate frame detection algorithm, allows the robot to learn the necessary skills for a multi-step task from a small number of demonstrations and generalize them to novel situations.

4.1 Introduction

A simple system that allows end-users to intuitively program robots is a key step in getting robots out of the laboratory and into the real world. Although in many cases it is possible for an expert to successfully program a robot to perform complex tasks, such programming requires a great deal of knowledge, is time-consuming, and is often task-specific. In response to this, much recent work has focused on robot learning from demonstration (LfD) [4], where non-expert users can teach a robot how to perform a task by example. Such demonstrations eliminate the need for knowledge of the robotic system, and in many cases require only a fraction of the time that it would take an expert to design a controller by hand.

Ideally, an LfD system can learn to perform and generalize complex tasks given a minimal number of demonstrations without requiring knowledge about the robot. Much LfD research has focused on the case in which the robot learns a monolithic policy from a demonstration of a simple task with a well-defined beginning and end [1, 45, 78]. This approach often fails for complex tasks that are difficult to model with a single policy. Thus, structured demonstrations are often provided for a sequence of subtasks, or *skills*, that are easier to learn and generalize than the task as a whole, and which may be reusable in other tasks.

However, a number of problems are associated with segmenting tasks by hand and providing individual skill demonstrations. Since the most natural way to demonstrate a task is by performing it continuously from start to finish, dividing a task into component skills is not only time-consuming, but often difficult—an effective segmentation can require knowledge of the robot's kinematic properties, internal representations, and existing skill competencies. Since skills may be repeated within and across tasks, defining skills also requires qualitative judgements about when two segments can be considered a single skill, or in deciding the appropriate level of granularity at which to perform segmentation. Users cannot be expected to manually manage this collection of skills as it grows over time.

For this reason, recent work has aimed at automating the segmentation process [17, 21, 40, 51, 52, 56]. Collectively, this body of work has addressed four key issues that are critical to any system that aims to learn increasingly complex tasks from unstructured demonstrations. (By *unstructured*, we refer to demonstrations that are unsegmented, possibly incomplete, and may originate from multiple tasks or skills.) First, the robot must be able to recognize repeated instances of skills and generalize them to new settings. Second, segmentation should be able to be performed without the need for *a priori* knowledge about the number or structure of skills involved in a task. Third, the robot should be able to identify a broad, general class of skills, including object manipulation skills, gestures, and goal-based actions. Fourth, the representation of skill policies should be such that they can be improved through practice.

Although many of these issues have already been addressed individually, no system that we are aware of has jointly addressed them all in a principled manner. Our contribution is a framework that addresses all of these issues by integrating a principled Bayesian nonparametric approach to segmentation with state-of-the-art LfD techniques as a first step towards a natural, scalable system that will be practical for deployment to end users. Segmentation and recognition are achieved using a Beta-Process Autoregressive HMM [32], while Dynamic Movement Primitives [45] are used to address LfD, policy representation, and generalization.

The combination and extension of these techniques allows a robot to segment and identify repeated skills in unstructured human demonstrations, create baseline skill policies from demonstration segments, leverage previous skill learning, and expand its skill library as needed. This approach is a first step toward a fully integrated LfD system that enables the long-term scaling up of task learning through the management, extension, and reuse of a growing skill library. We demonstrate this approach with experimental results using the PR2 mobile manipulator on both a simulated and physical task.

4.2 Background

4.2.1 Bayesian Nonparametric Time Series Analysis

Hidden Markov models (HMMs) are generative Bayesian models that have long been used to make inferences about time series data. An HMM models a Markov process with discrete, unobservable hidden states, or modes[1], which generate observations through mode-specific emission distributions. A transition function describes the probability of each mode at time $t+1$ given the mode at time t, but observations are limited to being conditionally independent given the generating modes. Given a set of observations, the forward-backward and Viterbi algorithms can be used to efficiently infer parameters for the model and determine the most likely sequence of modes that generated the data. Unfortunately, the number of modes must be specified *a priori* or chosen via model selection, which is prone to overfitting. This severely limits the usefulness of HMM inference when dealing with unstructured data. However, recent work in Bayesian nonparametrics offers a principled way to overcome these limitations.

The Hierarchical Dirichlet Process HMM (HDP-HMM) [99] uses a Hierarchical Dirichlet process prior over the transition and emission distributions of an HMM. The HDP-HMM allows for a potentially infinite number of latent modes to be discovered in a fully Bayesian manner, without the risk of overfitting or the need for prior knowledge. Since an HDP-HMM models an underlying Markov process, the probability of staying in the same mode for n consecutive steps is multiplicative, implying geometric mode durations, which can cause rapid mode switching. This assumption is incorrect for many real-world data sets, motivating the Sticky HDP-HMM [31], which adds a learned self-transition bias for each mode. An HDP-HMM also assumes that a mode emits observations that are independent and identically distributed, but this is clearly not true in the case where the observations exhibit temporal dependencies. Autoregressive HMMs [83] have been proposed to represent the these dependencies by adding causal links between observations in the HMM, but it can be difficult to represent the transition structure when the observation space is large or continuous.

[1]We refer to hidden states as *modes*, as to not confuse them with the RL concept of states.

The Beta Process Autoregressive HMM (BP-AR-HMM) [32] is a recent model that integrates many of the advantages of the aforementioned models, while extending them and providing principled solutions to some of their shortcomings. Like the Sticky HDP-HMM, it learns duration distributions from data. Improving on the HDP-HMM mechanism for sharing modes across time series, the BP-AR-HMM uses a beta process prior that leverages an infinite feature-based representation, in which each time series can exhibit a subset of the total number of discovered modes and switch between them in a unique manner. Thus, a potentially infinite library of modes can be constructed in a fully Bayesian way, in which modes are flexibly shared between time series, and an appropriate number of modes is inferred directly from the data, without the need for model selection. The BP-AR-HMM is also autoregressive and can describe temporal dependencies between continuous observations as a Vector Autoregressive (VAR) process, a special case of a linear dynamical system (LDS). A more complete description of the BP-AR-HMM can be found in Section 2.3.4.

4.2.2 Dynamic Movement Primitives

Dynamic Movement Primitives (DMPs) [45] provide a framework in which dynamical systems can be described as a set of nonlinear differential equations in which a linear point attractive system or limit cycle oscillator is modulated by a nonlinear function. Stability and convergence are guaranteed by introducing an additional canonical system, governed by linear equations that control a 0 to 1 phase variable that attenuates the influence of the nonlinear function over time. DMPs provide simple mechanisms for LfD, RL policy improvement, and execution, which scale easily in time and space and can support discrete or oscillatory movements [90]. In this chapter, we focus on the use of point attractive systems for implementing discrete movements with DMPs.

A discrete movement DMP can be described by the transformation system,

$$\tau\,\dot{v} = K(g-x) - Dv - K(g-x_0)s + Kf(s) \tag{4.1}$$

$$\tau\,\dot{x} = v, \tag{4.2}$$

and the canonical system,

$$\tau\,\dot{s} = -\alpha s, \tag{4.3}$$

for spring constant K, damping constant D, position x, velocity v, goal g, phase s, temporal scaling factor τ, and constant α [77]. The nonlinear function f can be represented as a linear combination of basis functions $\psi_i(s)$, scaled by the phase variable, s:

$$f(s) = \sum_{i=1}^{N} w_i \psi_i(s) s. \tag{4.4}$$

We use the univariate Fourier basis [53] for our function approximator, though others have commonly used normalized radial basis functions [77]. The spring and damping constants can be set to ensure critical damping, but we still must find appropriate weights w_i for the nonlinear function f.

Given a demonstration trajectory $x(t)$, $\dot{x}(t)$, $\ddot{x}(t)$ with duration T, we can use LfD to learn a set of values for these weights [90]. Rearranging equation 4.1, integrating equation 4.3 to convert time to phase, and substituting in the demonstration for the appropriate variables, we get:

$$f_{\text{target}}(s) = \frac{\tau^2\ddot{x} + \tau D\dot{x}}{K} - (g-x) + (g-x_0)s. \tag{4.5}$$

Setting the goal to $g = x(T)$, and choosing τ such that the DMP reaches 95% convergence at time $t = T$, we obtain a simple supervised learning problem to find the weights w_i for the basis functions. We use standard linear regression for this task.

This LfD procedure provides us with weights for a baseline controller that can be further improved through practice using RL [90], though we do not do so.

Several experiments have shown DMPs to be a promising solution to problems of control from both a computational and neuroscience perspective. DMPs have successfully been used to teach high-DOF robots complex tasks such as bipedal walking [90], an oscillatory tennis ball bouncing task [89], and skills for billiards [78]. In addition to the computational successes and theoretical guarantees of DMPs, there is experimental evidence that DMPs may provide a compelling account of human behavioral and neurological data. DMPs have been used as a model to explain and overturn the decades-old "2/3 power law" and the theory of planar piecewise movement segmentation in humans and primates [89]. Other experiments have shown that DMPs exhibit similar dynamical solutions for a ball bouncing task to those of human subjects and provide an improved explanation of the coupling structure of discrete and rhythmic motions [89]. Thus, DMPs provide a powerful framework in which to model and understand skill representation in both robot and human actors.

4.3 Learning from Unstructured Demonstrations

We now introduce a framework which integrates four major capabilities critical for the robust learning of complex tasks from unstructured demonstrations [75]. First, the robot must be able to recognize repeated instances of skills and generalize them to new settings. Given a set of demonstrations for a task, we use the BP-AR-HMM to parse the demonstrations into segments that can be explained by a set of latent skills, represented as VAR processes. The BP-AR-HMM enables these skills to be shared across demonstrations and tasks by employing a feature-based representation in which each skill corresponds to a feature that may or may not be present in a particular trajectory. Furthermore, this representation allows each trajectory to transition between skills in a unique manner, so that skills can be identified flexibly in a variety of situations, while still retaining globally shared properties.

Segmentation of trajectories into VAR models allows for tractable inference over the time-series dependencies of observations and provides a parameterization of each skill so that repeat instances can be recognized. This representation models how state changes over time, based on previous state values, potentially allowing instances of the same underlying skill to be recognized, even when performed with respect to different coordinate frames. The BP-AR-HMM also models skill-dependent noise characteristics to improve the identification of repeated skills. By recognizing repeated skills, a skill library can be incrementally constructed over time to assist in segmenting new demonstrations. Additionally, skill controllers that have been previously learned and improved through practice can be reused on new tasks. Thus, recognition of repeated skills can reduce the amount of demonstration data required to successfully segment and learn complex tasks. Similarly, if we have multiple examples of a skill, we can discover invariants that allow us to generalize the skill to new situations robustly. In this chapter, we use these data to identify the coordinate frames that each skill takes place in, as described in detail in the next section.

Second, segmentation must be able to be performed without the need for *a priori* knowledge about the number or structure of skills involved in a task. The BP-AR-HMM places a beta process prior over the matrix of trajectory-feature assignments, so that a potentially infinite number of skills can be represented; the actual finite number of represented skills is decided upon in a principled, fully Bayesian way. Skill durations are modeled indirectly through a learned self-transition bias, preventing skills from being over-segmented into many small components unnecessarily. The BP-AR-HMM also provides reliable inference, having only a few free parameters that are robust to a wide range of initial settings and hyperparameters that conform to the data as inference progresses. Thus, little tuning should be necessary for varying tasks for a given robotic platform.

Third, our system must be able to identify a broad, general class of skills. Since our segmentation method is based upon state changes, rather than absolute state values, we are able to identify a wide array of movement types ranging from object manipulation skills to gestures and goal-based actions. Furthermore, by identifying the relevant coordinate frame of repeated skills, we can discover specific objects and goals in the world that skills are associated with.

Fourth, the representation of skill policies should be such that they can be easily generalized and improved through practice. To accomplish this, we represent skill controllers in the DMP framework. The spring-damper mechanics of a DMP allow for easy generalization, since the start and goal set-points can be moved, while still guaranteeing convergence and maintaining the "spirit" of the demonstration through the output of the nonlinear function. In addition to affording a simple LfD algorithm, the linear function approximator used to represent the nonlinear forcing function allows for skill improvement through practice by modifying the weights through an RL algorithm such as the Natural Actor-Critic [81], though we do not do so in this work.

4.4 Methodology

4.4.1 Demonstrations

For the first two experiments in the following section, we use a simulated Willow Garage PR2 mobile manipulator and the ROS framework; the final experiment uses a real PR2. The PR2 personal robot [107] is a two-armed robot platform with an omnidirectional base designed for mobile-manipulation tasks. The PR2 is equipped with 7-degrees of freedom compliant arms. The PR2 has a sensor suite useful for mobile manipulation tasks, including a tilting laser scanner attached to the head, two stereo cameras, a Microsoft Kinect, a laser scanner mounted on the base, and a body mounted IMU. By design, the PR2 does not have to deal with the many degrees of freedom that come along with legs or complex hands, making it easier to design interfaces, create controllers, and interact with the robot.

We used hand-coded controllers to provide task demonstrations to the simulated robot. The robot is placed in a fixed position in front of a table, as shown in Figure 4.1. At the beginning of each demonstration, the robot looks downward and captures a stereo image of the table. It then removes the flat table top and obtains a point cloud for each of the objects on the table, recording their positions and dimensions. On the real robot, object positions are determined by a visual fiducial placed on each object of interest. Once the demonstration begins, data are collected by recording the 7 joint angles in the left arm and the gripper state (a scalar indicating its degree of closed-ness). Offline, the joint angles are converted to a series of 3D Cartesian positions and 4D quaternion orientations, which are subsampled down to 10 Hz and smoothed, along with the gripper positions.

4.4.2 Segmentation

We build on a BP-AR-HMM implementation made available by Emily Fox[2] to segment sets of demonstration trajectories. We preprocess the demonstrations so that the variance of the first differences of each dimension of the data is 1, as in Fox et al. [33], and adjust it to be mean zero. We choose an autoregressive order of 1 and use identical parameters as those used by Fox on a human exercise motion capture dataset [33], with one exception—in the simulated experiments, we adjust the matrix-normal inverse-Wishart prior on the dynamic parameters, since the simulated data have significantly different statistical properties from that in Fox et al. [33]. To segment the demonstrations, we run the combined Metropolis-Hastings and Gibbs

[2]`http://stat.wharton.upenn.edu/~ebfox/software`

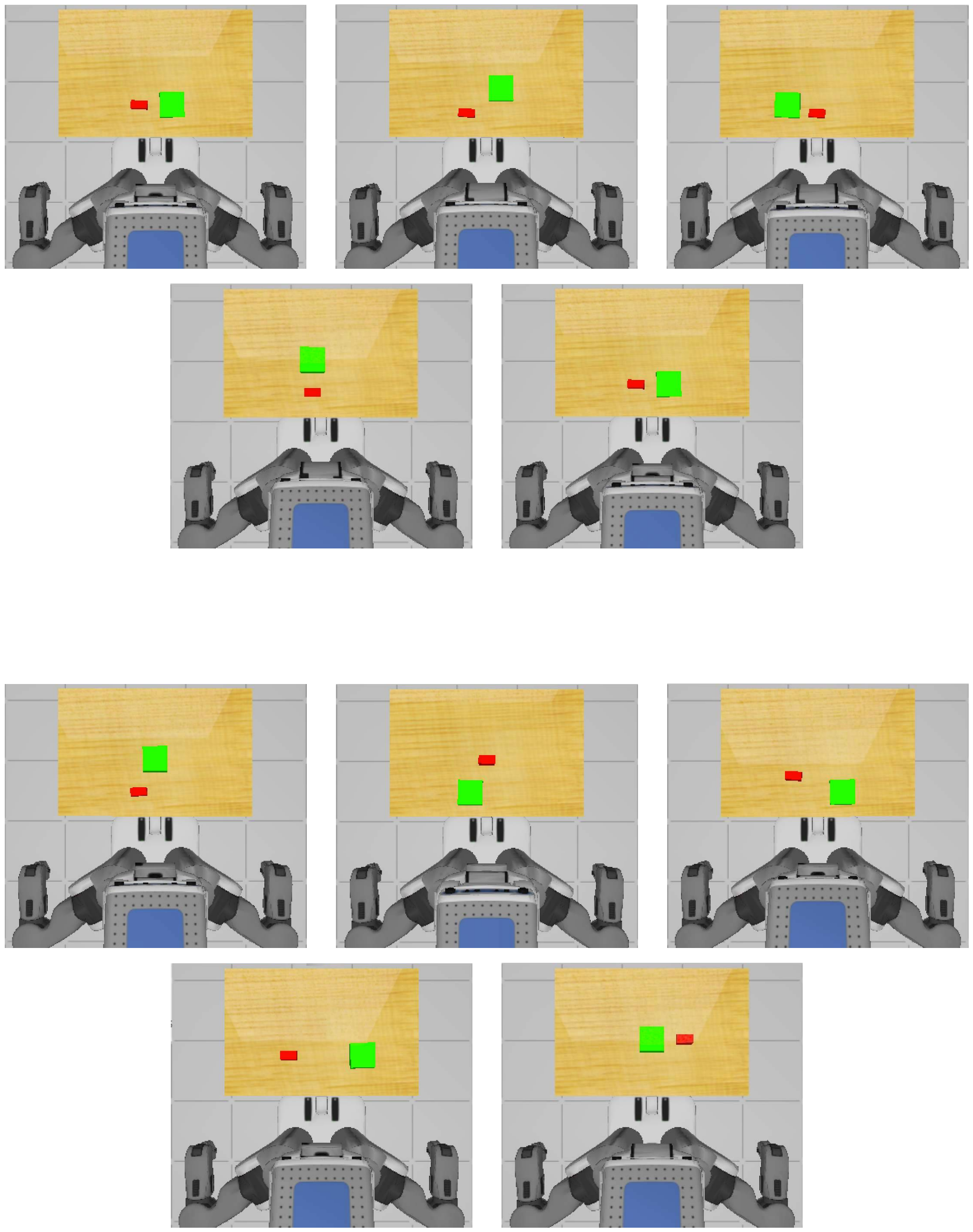


Figure 4.1. A top-down view of the robot and table layout for 5 task demonstration configurations (top) and 5 novel test configurations (bottom).

sampler 10 times for 1000 iterations each, producing 10 segmentations. Qualitatively, the segmentations across runs were very consistent, but to ensure good results, the segmentation from the 10 runs with the highest log likelihood of the feature settings is selected.

4.4.3 Coordinate Frame Detection

After the demonstrations are segmented, each segment is examined to infer the coordinate frame that it is occurring in. Even though segments assigned to the same skill correspond to similar movements, they may be happening in different frames of reference. For example, a repeated reaching motion may be classified as being generated by the same skill, but be reaching toward several different objects. In order to robustly replay tasks in novel configurations, it is desirable to determine which coordinate frame each segment is associated with, so that DMP goals can be generalized correctly.

We define a coordinate frame centered on each known object, along with one centered at the torso of the robot. Other frames could be used as well if desired, such as a frame relative to the gripper, or a world frame. Then, the final point of each segment is plotted separately in each of the coordinate frames, and clusters are found in each frame by identifying points within a Euclidean distance threshold of each other. The reasoning is that clusters of points indicate that multiple segments have similar endpoints in a particular coordinate frame, suggesting that the skill often occurs in that frame of reference.

After the points are clustered in each frame, all the singleton clusters are discarded. If any remaining segment endpoint belongs only to a cluster in a single coordinate frame, then the evidence is unambiguous, and that segment is assigned to that coordinate frame. Otherwise, if a segment endpoint belongs to clusters in multiple frames, it is simply assigned to the frame corresponding to the largest clus-

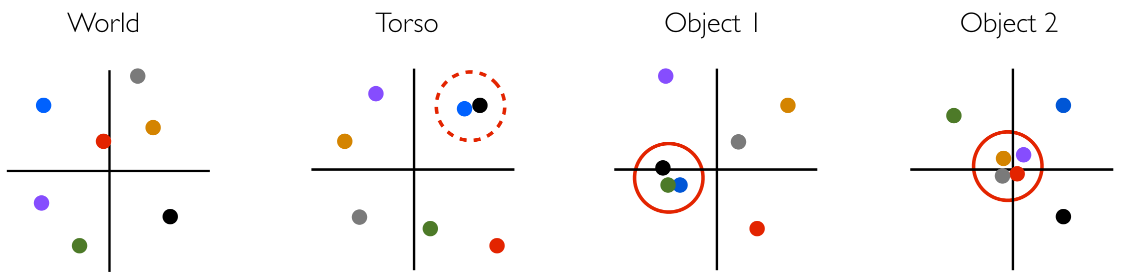


Figure 4.2. An illustrative example showing the endpoints of seven trajectories plotted with respect to four different coordinate frames. The solid circles denote detected clusters, while the dotted circle indicates a candidate cluster that is rejected as noise, since all associated points can be assigned to a potentially larger cluster. In our experiments, this clustering happens in three dimensions, rather than two as illustrated here.

ter. Figure 4.2 shows an example of this clustering process. It should be emphasized that any coordinate frame inference method could be used in place of ours, and many other skill invariants could be exploited. The purpose of this method is primarily to demonstrate the utility of being able to segment and recognize repeated skills.

4.4.4 Task Replay

To perform a task in a novel configuration, we first determine the poses and identities of objects in the scene, using either stereo data (simulated experiment) or visual fiducials (real robot). The position of each object is then examined to find the demonstration that begins with the objects in a configuration that is closest to the current one in a Euclidean sense. We only consider demonstrations that have an identified coordinate frame for every segment, so that the task will generalize properly. A DMP is then created and trained using the LfD algorithm from section 4.2.2 for each segment in the demonstration. However, rather than using the final point of a segment as the goal of a DMP, each goal is adjusted based on the coordinate frame that the segment takes place in. If the segment is associated with the torso frame, it requires no adjustment. Otherwise, if it is associated with an object frame, the goal is adjusted by the difference between the object's current position and its position in the demonstration. Finally, the DMPs are executed in the sequence specified by the demonstration. A plan is generated by each of the DMPs until the predicted state is within a small threshold of the goal. Each plan is a Cartesian trajectory (plus a synchronized gripper state) that is converted into smooth joint commands using inverse kinematics and spline interpolation. Algorithm 4 provides a summary of our method, along with the graphical overview shown in Figure 4.3.

Algorithm 4 Learning from Unstructured Demonstrations

Input: Demonstration trajectories $\mathbf{T} = (T_1, \ldots, T_N)$, corresponding object pose observations $\mathbf{O} = (O_1, \ldots, O_N)$, and current object poses $O_{current}$

```
1. Segment the demonstrations with the BP-AR-HMM:
     (segments, labels) = $Segmentation(\mathbf{T})$
2. Find coordinate frame assignments:
     coord_frames = $CoordinateFrameDetection$(segments, labels, $\mathbf{O}$)
3. Learn params of a DMP for each segment:
     dmps = $LearnDmpParams$(segments, coord_frames)
4. Execute sequence of DMPs corresponding to the segment ordering of the most
   similar demonstration:
     demo_num = $NearestFirstObservation(O_{current}, \mathbf{T})$
     $ExecuteSequence$(dmps, demo_num)
```

4.5 Experiments

4.5.1 Experiment 1: Pick and Place (Simulated)

The first experiment demonstrates the ability of a robot in our framework to learn and generalize a complex task by segmenting multiple task demonstrations, identifying repeated skills, and discovering appropriate segment reference frames. Each instance of the task begins with two blocks on the table—a smaller red block and a larger green block. The robot always starts in a "home" configuration, with its arms

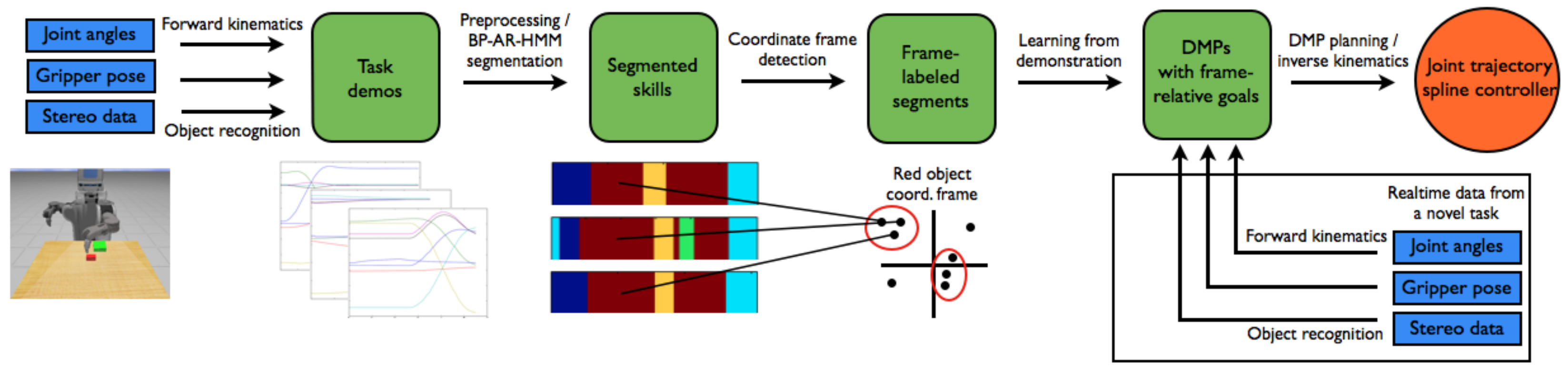

Joint angles
Gripper pose
Stereo data
Forward kinematics
Object recognition
Task demos
Preprocessing / BP-AR-HMM segmentation
Segmented skills
Coordinate frame detection
Frame-labeled segments
Red object coord. frame
Learning from demonstration
DMPs with frame-relative goals
DMP planning / inverse kinematics
Joint trajectory spline controller
Realtime data from a novel task
Forward kinematics
Joint angles
Gripper pose
Stereo data
Object recognition

Figure 4.3. Overview of the framework used in the experiments, as described in section 4.4

at its sides so that its field of view is unobstructed. We provide 5 task demonstrations for 5 different configurations of the blocks, as shown in the first row of Figure 4.1 (configurations 1 and 5 are identical, but the demonstration is performed at a higher speed in the latter configuration). In each demonstration, the robot first picks up the red block, returns to the home position, places the red block on the green block, and returns to the home position once more.[3]

Figure 4.4 shows the results of segmentation. The top row shows one colored bar per skill, while the bottom row displays the skill divisions overlaid on a plot of each of the 8 dimensions of the demonstration data. The BP-AR-HMM consistently recognizes repeated skills across demonstrations, even though they occur at differing speeds and with different goals. The segmentations are highly similar, with the exception of the second demonstration, which identifies one additional skill that the others do not have. It is worth noting that despite the extra skill being inserted in the segmentation, the rest of the segmentation is essentially the same as the others. This is a direct benefit of the BP-AR-HMM allowing each trajectory to have its own switching dynamics, while sharing global features.

Next, we examine task generalization to 5 novel test configurations, shown in the bottom row of Figure 4.1, to determine whether our segmentation produced semantically meaningful results. Our method was able to successfully identify a coordinate frame for every segment except the extra segment in demonstration two (which is impossible to infer, since there is only one example of it). Using this information, the robot performed task replay as described in section 4.4.4. In all 5 novel configu-

[3]Due to the planning delay in the hand written controllers there are some pauses between segments which we remove to avoid giving the segmentation algorithm an unfair advantage.

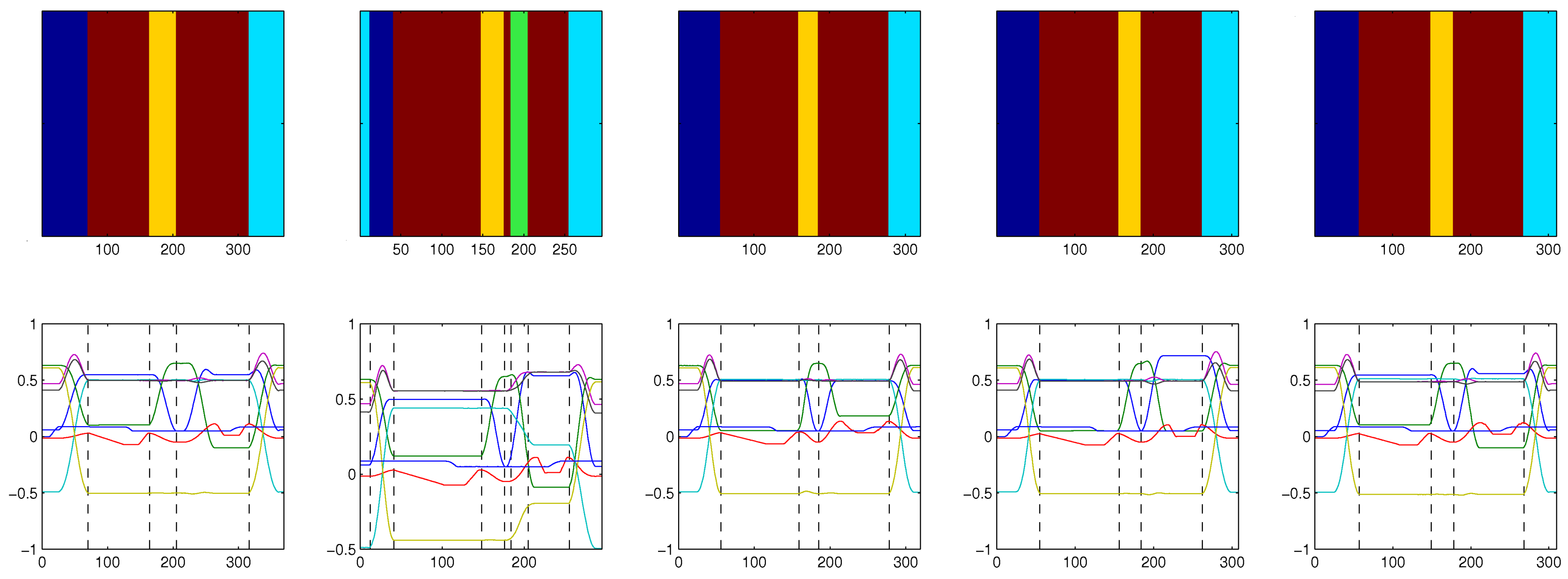
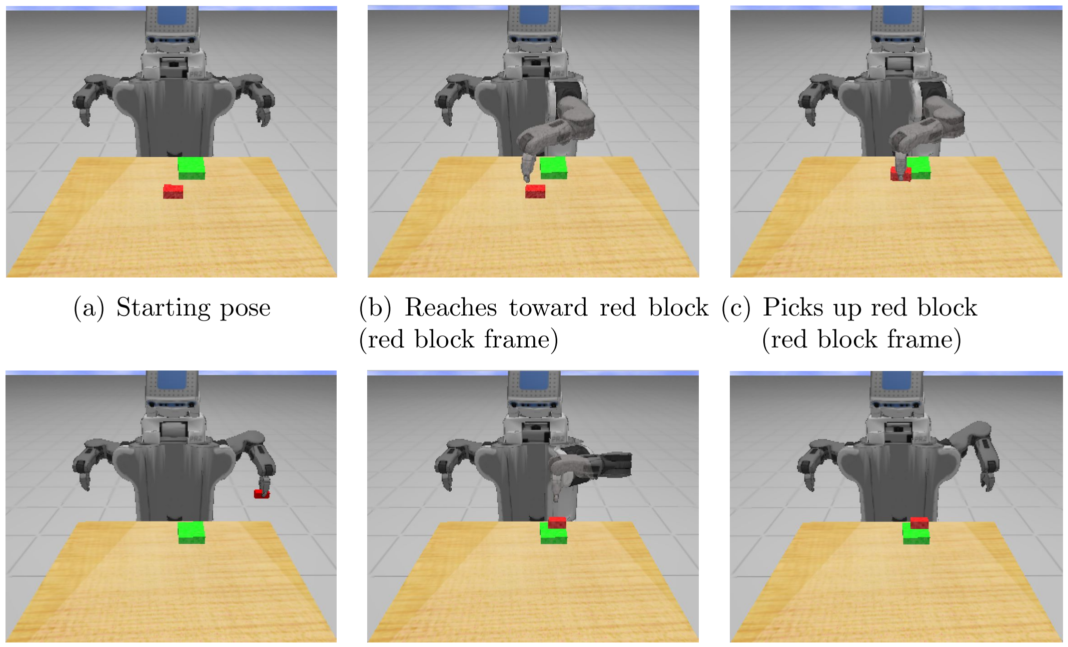


Figure 4.4. Top: BP-AR-HMM segmentations of the 5 demonstration trajectories for the pick and place task. Time (in tenths of a second) is shown on the x-axis. Skill labels at each time step are indicated by unique colors. Bottom: Segmentation points overlaid on the demonstrated 8D movement data.



(a) Starting pose

(b) Reaches toward red block (red block frame)

(c) Picks up red block (red block frame)

(d) Returns to home position (torso frame)

(e) Places red block on green block (green block frame)

(f) Returns to home position (torso frame)

Figure 4.5. Successful task replay on a novel test configuration for the pick and place task, demonstrating generalization. From left to right: the starting pose and the final point of each executed DMP. Automatically detected coordinate frames used for each segment are listed in parentheses.

rations, the robot was able to successfully generalize and place the red block on the green block.[4]

Figure 4.5 shows the starting state of the robot and the resulting state after each DMP is executed in a novel test configuration. Here, it becomes clear that the results of both the segmentation and coordinate frame detection are semantically intelligible. The first skill is a reaching skill to right above the red block. The second skill moves down, grasps the red block, and moves back upward. The third skill goes back to the home position. The fourth skill reaches toward the green block, moves downward,

[4]The green block in novel configuration 4 was partially out of the robot's visual range, causing part of it to be cut off. Thus, it placed the red block too close to the edge of the green block, causing it to tumble off. However, given the available information, it acted correctly.

releases the red block and moves back upward. Finally, the fifth skill goes back to the home position. Notice that the second and fourth segments are identified by the BP-AR-HMM as being the same skill, despite having different relevant coordinate frames. However, in both skills, the arm moves down toward an object, changes the gripper pose, and moves back upward; the reach from the home position toward the green block gets rolled into this skill, rather than getting its own, seemingly because it is a smoother, more integrated motion than the reach and grasp associated with the red block.

Given the commonality of pick and place tasks in robotics, success in this domain may seem trivial. However, it is important to keep in mind that the robot is given only demonstrations in joint space and absolutely no other *a priori* knowledge about the nature of the task. It does not know that it is being shown a pick and place task (or doing grasping at all). It is unaware of the number of subtasks that comprise the task and whether the subtasks will be object-related, gestural, or have other sorts of objectives. Beginning with only motion data and a simple assumption about the types of coordinate frames that are relevant to inspect, the robot is able to automatically segment and generalize a task with multiple parts, each having its own relevant coordinate frame.

4.5.2 Experiment 2: Using a Skill Library (Simulated)

The first experiment demonstrated that our method can learn and generalize a complex task when given a sufficient number of demonstrations. However, this type of learning will not scale up to more complex tasks easily unless the robot can incrementally build a library of skills over time that allows it to quickly recognize previously seen skill / coordinate frame combinations and reuse complex skill controllers that have been improved through practice. To demonstrate our system's capability to recognize skills in this manner, we simulate a previously existing library of skills by providing the robot with a pre-segmented demonstration of the previous experiment. We then give it a single demonstration of the task to see if it can segment it using the “library” of skills.

The BP-AR-HMM correctly recognized each of the skills in the task as being a skill from the pre-existing library. Thus, assuming the robot already had learned about these skills from previous experiences, it would allow a user to provide only a single demonstration of this task and have the robot correctly segment and generalize the task to new configurations. This serves as a proof-of-concept that our proposed framework has the right basic properties to serve as a building block for future models that will scale up LfD to more complex tasks than have previously been possible. It also emphasizes that our method can learn tasks from unstructured demonstrations, as the majority of demonstrations were not even of the task in question, but of a sub-component, unbeknownst to the robot. This is a significant result, as it suggests that a similar technique could be used to incrementally build a growing library of skills over time that could be leveraged to perform one-shot learning of complex tasks from demonstration.

4.5.3 Experiment 3: The Whiteboard Survey (Physical PR2)

Finally, we demonstrate that our method is scalable to a real robot system, using a physical PR2. Figure 4.6 shows one configuration of a task in which the PR2 must fill out a survey on a whiteboard by picking up a red marker and drawing an ’X’ in the checkbox corresponding to “robot” while ignoring the checkboxes for “male” and “female”. Each checkbox has its own unique fiducial placed one inch to the left of it, while the container that holds the marker has a fiducial directly on its front. The positions of the checkboxes and the marker container on the whiteboard, as well as the position of the whiteboard itself, change between task configurations. Two kinesthetic demonstrations in each of three task configurations (various arrangements of the

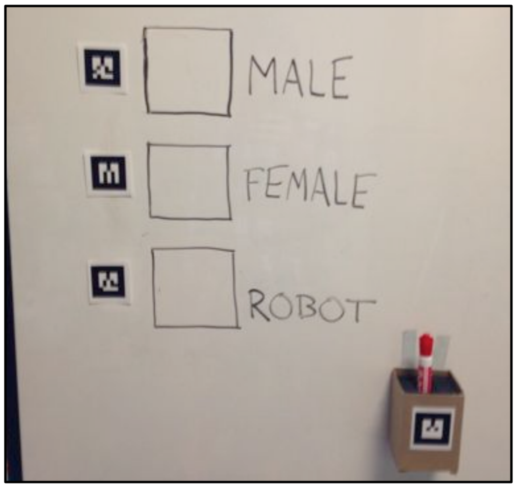


Figure 4.6. Example configuration of the whiteboard survey task.

checkboxes and marker box) were provided, along with one additional demonstration in which the marker is picked up and then lifted above the robot's head. *Kinesthetic* refers to demonstrations in which the robot is physically manipulated by the user to perform the task. An example demonstration is shown in Figure 4.7.

Figure 4.8 shows that the BP-AR-HMM generally parses the demonstrations into three main segments, corresponding to reaching for the marker, grasping and lifting the marker, and drawing an 'X' in the checkbox. However, the reaching and drawing segments are considered to be the same skill. This appears to happen because both motions are statistically similar, not in terms of absolute position, but in the way that the positions evolve over time as a VAR system. Our coordinate frame detection successfully disambiguates these skills and splits them into two separate skill/coordinate frame combinations. Demonstrations 1, 2, and 5 contain a small additional skill near the beginning that corresponds to a significant twitch in the shoulder joint before any other movement starts, which appears to correspond to the teacher's first contact with the arm, prior to the demonstration. Finally, although the last demonstration is of a different task, the reaching and grasping/lifting skills are still successfully recognized, while the final motion of lifting the marker over the

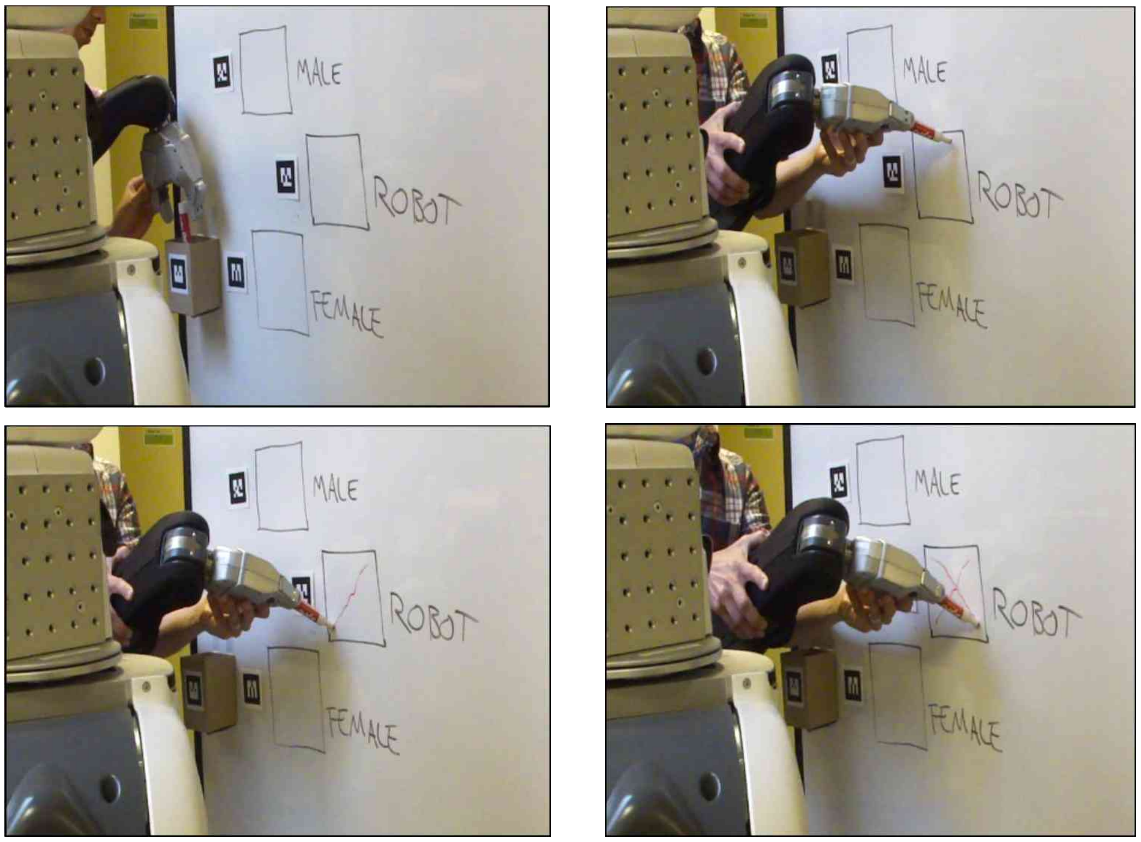


Figure 4.7. A kinesthetic demonstration of the whiteboard survey task.

robot's head is given a unique skill of its own. Despite having only a single example of the over-head skill, the BP-AR-HMM robustly identified it as being unique in 50 out of 50 trial segmentations, while also recognizing other skills from the main task. After the learning phase, the robot was able to successfully replay the task in three novel configurations, an example of which is shown in Figure 4.9.

4.6 Discussion

In this chapter, we presented a novel method for segmenting demonstrations, recognizing repeated skills, and generalizing complex tasks from unstructured demonstrations. Though previous research has addressed many of these issues individually, our method aims to address them all in a single integrated and principled framework.

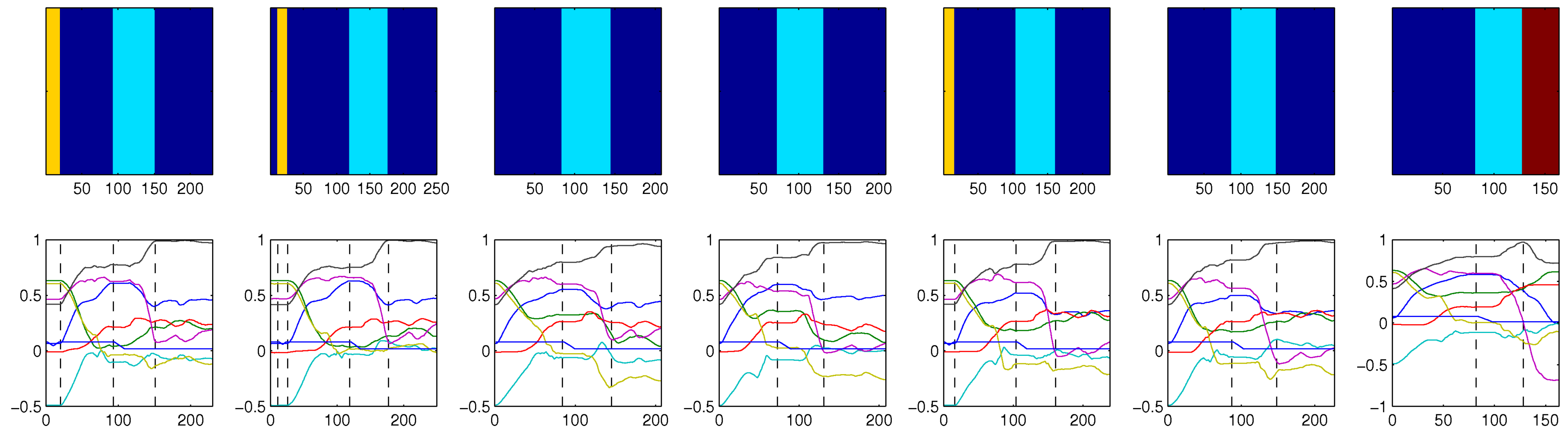
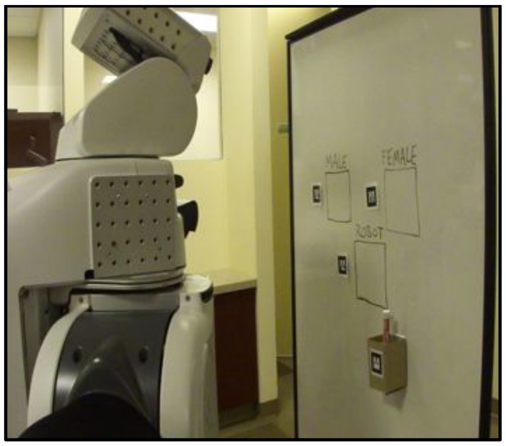
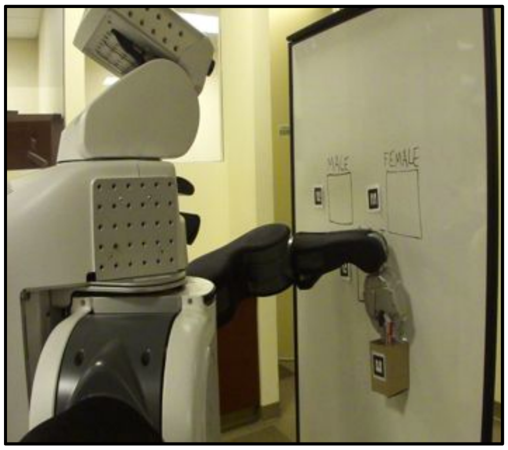


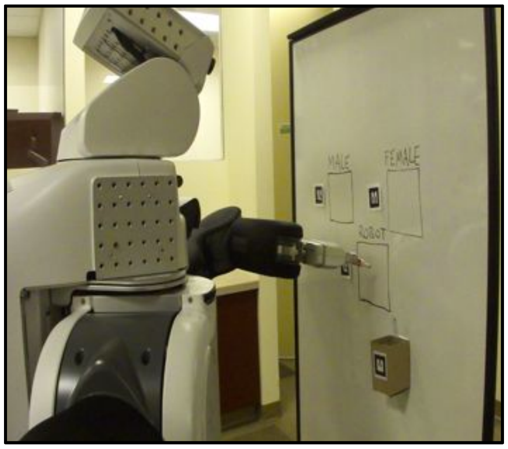
Figure 4.8. Top: BP-AR-HMM segmentations of the 7 demonstration trajectories for the whiteboard survey task. Time (in tenths of a second) is shown on the x-axis. Skill labels at each time step are indicated by unique colors. Bottom: Segmentation points overlaid on the demonstrated 8D movement data.



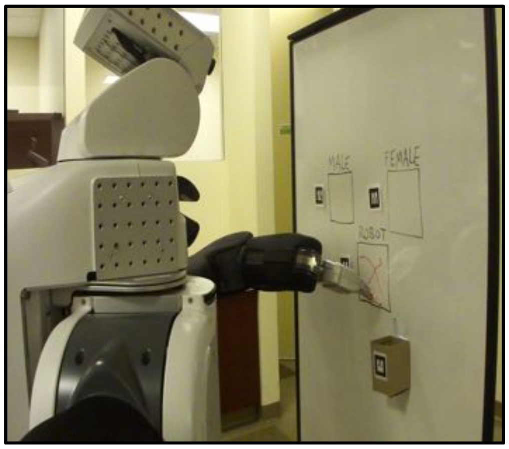
(a) Starting pose



(b) Reaches for marker (marker frame)



(c) Grasps marker and lifts toward checkbox (robot checkbox frame)



(d) Draws 'X' (robot checkbox frame)

Figure 4.9. Successful task replay on a novel test configuration for the whiteboard survey task, demonstrating generalization. From left to right: the starting pose and the final point of each executed DMP. Automatically detected coordinate frames used for each segment are listed in parentheses.

By using the BP-AR-HMM and DMPs, we are able to experimentally learn and generalize a multiple step task on the PR2 mobile manipulator and to demonstrate the potential of our framework to learn a large library of skills over time.

Our framework demonstrates several of the critical components of an LfD system that can incrementally expand a robot's competencies and scale to more complex tasks over time. However, this work is only a first step toward such a system, and leaves a great number of directions open for future research. A more nuanced method must be developed for managing the growing library of skills over time, so that inference in our model does not become prohibitively expensive as the size of the library grows; currently inference in the BP-AR-HMM takes several minutes for only 7 demonstrations on a laptop with a 2.2 GHz Intel quad-core i7 processor and 8 GB of RAM. One possible way to mitigate this in the future is to "freeze" some skills in the library, such that they can be recognized in new demonstrations, but are no longer candidates for modification during inference.

While our model allows for DMP policy improvement through RL, we did not address such improvement experimentally in this chapter. Future work may use techniques such as inverse RL [1] to derive an appropriate reward function for each skill so that policy improvement can be effectively applied. A learned reward (cost) function could also be used to in conjunction with a motion planner as a more adaptive alternate to DMP motion generation.

There are also many more opportunities to take advantage of abstractions and invariants in the data; searching for skill coordinate frames is a very simple example of a much richer class of generalization techniques. For example, in this chapter, we constructed DMPs from and ordering of single segments that came from the task configuration most similar to the current one that the robot faces. This is a rather inflexible way to sequencing skills that can lead to failure in cases where errors might be made during execution or where adaptive behavior is required. In the next chapter, we examine more intelligent methods for skill sequencing that can be applied to make better use of the demonstration data that we have available. By leveraging additional structure, the robot can learn to classify and sequence skills adaptively based on observations and learn through interactive corrections how to deal with contingencies encountered during execution.

CHAPTER 5
INCREMENTAL SEMANTICALLY GROUNDED SKILL DISCOVERY

In the previous chapter, a novel system was introduced to segment demonstrations into motion categories, find appropriate coordinate frames for these motions, and create skill controllers for each category using DMPs. However, task replay consisted of essentially blind replay of a sequence of skills that had been seen previously, which can lead to brittle performance in many tasks. More complex tasks often have multiple valid execution paths, contingencies that may arise during execution, or exhibit partial observability and/or perceptual aliasing. We introduce a novel method to further subdivide discovered motion categories into semantically grounded states in a finite-state representation of a task, enabling intelligent, adaptive replay. Performance can then be incrementally improved through interactive corrections that provide additional data where they are most needed. Together, this allows for intelligent discovery and sequencing of semantically grounded skills to create flexible, adaptive behavior that can be improved through natural interactions with the robot.

5.1 Introduction

Automatic segmentation and recognition of repeated structure in a task makes learning more tractable, enables data reuse and transfer, and can reveal exploitable invariants in the data, as shown in the previous chapter. However, in all but the simplest of tasks, a fully autonomous, flexible system requires the automated sequencing of these primitives to perform a task, rather than rote execution in a fixed order.

Current LfD methods take a variety of approaches to sequencing, ranging from associative skill memories [79], in which all primitives are made available for selection at each decision point, to other schemes in which primitives are executed in a fixed order [2, 75]. We argue that a method between these two extremes is most effective, in which there is flexibility in choosing what primitive is scheduled next, but the choices are limited at each decision point to keep the discriminative task tractable as the number of primitives grow.

A novel method is presented to sequence automatically discovered primitives that makes minimal assumptions about the structure of the task and can sequence primitives in previously unseen ways to create new, adaptive behaviors. As in the previous chapter, a Beta Process Autoregressive Hidden Markov Model is used to segment continuous demonstration data into motion categories with associated coordinate frames. Tests are then performed on the motion categories to further subdivide them into semantically grounded movement primitives, in which exemplars of the motions are correlated with data about observed actions, objects, and their relationships. These grounded primitives are then used to create a finite-state representation of the task, in which each state has an associated set of exemplars of the relevant movement primitive, plus a trained classifier used to determine state transitions. The resulting finite-state automaton (FSA) can then be used to replay a complex, multi-step task [74].

However, initial demonstrations do not always cover all the possible contingencies that may arise during the execution of a task. When most LfD methods fail at task replay, the onus is on the user to design new demonstrations that can fill in the gaps in the robot's knowledge. Instead, a data-driven approach is introduced that provides additional data where they are most needed through interactive corrections. These corrections are provided by the user at the time of failure and are treated as additional demonstrations that can be segmented, used to improve the structure of the

FSA, and provide additional examples of relevant primitives. Together, this allows for iterative, incremental learning and improvement of a complex task from unsegmented demonstrations. The utility of this system is shown on a complex furniture assembly task using a PR2 mobile manipulator.

5.2 Constructing Finite-State Task Representations

When performing complex multi-step tasks, it is vital to not just have the right skills for the task, but also to choose and execute skills intelligently based on the current situation. Tasks cannot always be broken down into a simple linear chain of controllers; rather, many complex tasks require the ability to adaptively handle novel and changing situations in addition to unforeseen contingencies that may arise. Some recent work has focused on developing associative skill memories [79] to adaptively match sensory input with predefined skills to perform a task. However, such skills cannot always be known ahead of time, and more structure than pure associativity may be required when there is *perceptual aliasing* [39] in the task. By automatically segmenting primitives from data and constructing a finite-state representation of a task, it is possible to combine skill-associativity with a powerful state framework that enables robots to automatically discover appropriate skills, reuse data efficiently, adapt to novel situations, and handle perceptual aliasing.

A finite-state automaton (FSA) is a natural representation for modeling transitions between discrete primitives. By using DMPs at a low-level to represent movement primitives and an FSA for high-level decision making, the advantages of both can be gained, allowing the representation of virtually any continuous motion, while also being able to flexibly make critical choices at a small number of discrete decision points. To define an FSA that represents a task, a notion of states and transitions is required. A state in the FSA ideally corresponds to a semantically meaningful step or action in the task that implies that a particular primitive should be executed. Each state stores a list of exemplars of a particular primitive that have been observed from demonstration. Transitions are dictated by a mapping from observations to successor states, which can be implemented as a multi-class classifier.

However, segmentation of demonstrations via the BP-AR-HMM provides us with motion categories based on statistical properties of the movements, not semantically meaningful actions. For example, a small twist of the wrist required as part of a grasp can have a nearly identical VAR formulation as the continuous twisting required to screw a piece in during an assembly task; these movements have different semantic meanings, but similar statistical properties. So how can semantically grounded primitives be created from motion categories?

We make the assumption that exemplars of a semantically grounded primitive will generally fall into the same motion category (i.e. have similar statistical properties), but that all examples of a particular motion category will not necessarily belong to the same primitive. Following this logic, exemplars of motion categories can be split into multiple groups that correspond to grounded primitives by using semantically meaningful splitting criteria. This can be achieved by performing splits based on the correlation of state visitation history with other types of semantic information, such as the exemplar's coordinate frame, length, or successor state. The state visitation history of an exemplar (the states in the FSA that the previous segments in the demonstration trajectory belong to) is a critical piece of information for splitting, because it provides a notion of sequence—examples of a motion category may be repeated several times in a task, but may correspond to semantically different primitives that are appropriate during various phases of task progress. In this case, only parent states are examined, i.e. one-step histories, due to the relative sparsity of data in an LfD setting.

If a semantic difference between exemplars at a single state (such as the coordinate frame of the exemplar) can be predicted by the parentage of the exemplar, then splitting the state has several benefits. First, it helps to determine which exemplars should be considered in different phases of the task. Such a split reduces the number of possible transitions from a single state, removing some of the burden from the transition classifiers and mitigating perceptual aliasing, where perceptual information alone is not enough to correctly determine the next action. This also has the effect of minimizing the number of inappropriate exemplars that can be chosen at a particular state, while maximizing data reuse by only splitting when it is warranted by correlative evidence—splitting at every node with multiple parents would induce a very strong notion of sequence, but would over-segment the data into many states, each of which would have very few associated exemplars and could only be executed in an order that had previously been observed.

It is not expected that this process will always work on the first try, due to factors like unforeseen contingencies and lack of sufficient data. Thus, a way to incrementally improve the FSA structure and task performance is required. For this, a data-driven method of providing interactive corrections is used that allows the user to halt task execution at any time and provide a corrective demonstration of the remainder of the task. This provides additional data where they are most needed—situations in which intervention is required for successful execution—through a natural, intuitive mechanism. Corrections can be used to account for unanticipated situations that were not covered in the original demonstrations, contingencies like missing a grasp or dropping an object, or incorrect movement sequencing. These corrections can then be treated as additional demonstrations and jointly segmented with the rest of the existing data, producing an improved FSA structure with additional exemplars of relevant primitives. This iterative process can be repeated as needed to address shortcomings in performance as errors occur. Figure 5.1 shows an overview of the whole system, which is described in greater detail in the following section.

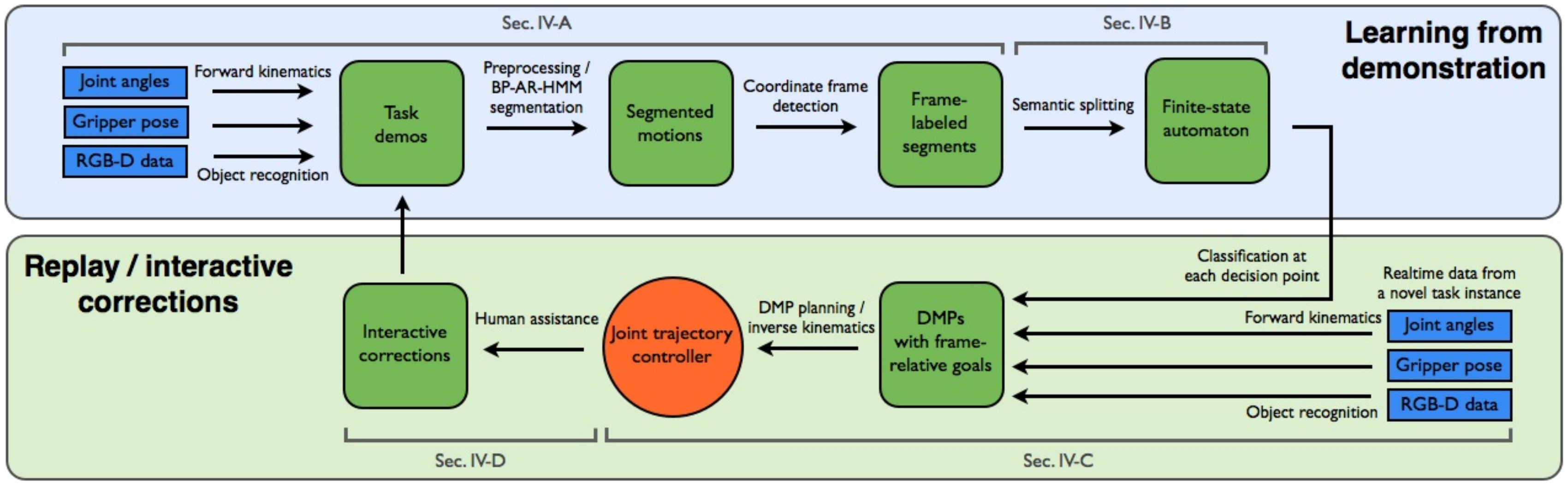


Figure 5.1. Overview of the iterative learning from demonstration framework.

5.3 Methodology

5.3.1 Demonstrations and segmentation

For all experiments, a PR2 mobile manipulator is used as the robotic platform. Task examples are provided to the robot via kinesthetic demonstrations, in which the teacher physically moves the arms of the robot to perform the task and uses a joystick to set the degree of closure of the grippers. AR tags, a type of visual fiducial, are used to track relevant pre-determined *task objects* using combined visual and depth data from a head-mounted RGB-D camera on the PR2.

During the i^{th} demonstration, the pose information $X_i = (x_{i,1}, \ldots, x_{i,\tau_i})$ with $x_{i,t} \in SE(3) \times SE(3) \times \mathbb{R}^2$ is recorded for each time t at 10 Hz, consisting of the Cartesian pose of the end effector plus gripper pose (1-D measure of openness) for both arms. The active arm can be changed by pressing a button on a joystick; the previously active arm becomes stiff, the inactive arm becomes loose and ready for interaction, and the arm switch event is recorded for later use. Additionally, a filtered estimate of the last observed Cartesian pose of all n task objects $O_i = (o_{i,1,1}, \ldots, o_{i,n,\tau_i})$, with $o_{i,j,t} \in SE(3)$ recorded for each object j at each time step t. At the beginning of the task, if not all relevant task objects are visible, the robot will prohibit the demonstration from starting and look around until an initial pose is known for all n objects.

After a set of m demonstrations $(X_1, O_1), \ldots, (X_m, O_m)$ of the task have been collected in various configurations, the robot pose information $X_1, \ldots, X_m$ can be segmented and labeled by the BP-AR-HMM using the same procedure and parameters as Niekum et al. [75]. However, this is done separately for data that comes from each arm, forcing segment breaks in locations where the active arm is switched and creates a separate library of motion categories for each arm.

The segmentation process provides a set of segment lists $S_1, \ldots, S_n$, such that $S_i = (s_{i,1}, \ldots, s_{i,q_i})$ and $concat(s_{i,1}, \ldots, s_{i,q_i}) = X_i$. Additionally, segmentation returns corresponding label vectors $L_1, \ldots, L_m$, such that $L_i = (l_{i,1}, \ldots, l_{i,q_i})$, where $l_{i,j} \in \mathbb{Z}$ is a label for the j^{th} segment in S_i, $s_{i,j}$. Each integer label corresponds to a unique motion category discovered by the BP-AR-HMM segmentation that is parameterized as a VAR process. Finally, the clustering method described in Niekum et al. [75] is used to automatically discover coordinate frame assignment lists $A_1, \ldots, A_n$ with $A_i = (a_{i,1}, \ldots, a_{i,q_i})$ and $a_{i,j} \in \{\text{'object 1'}, \ldots, \text{'object n'}, \text{'torso'}, \text{'none'}\}$.

5.3.2 FSA construction

Defining the set L^* as the union of all the unique labels from the segment label vectors $L_1, \ldots, L_m$, a finite-state automaton that represents the task can begin to be constructed by creating nodes[1] $N_1, \ldots, N_u$, with corresponding labels $L_1^*, \ldots, L_u^*$, where $u = |L^*|$. For $k \in \{1, \ldots, u\}$, each node N_k is assigned a set E_k of all exemplars $s_{i,j}$ that have the same label as N_k, and the label of the previous and next segments is also recorded (or START or END if there is no previous or next segment), which we denote as $prev(s)$ and $next(s)$. A $u \times u$ transition matrix T can then be constructed, where each entry $T_{a,b}$ is set to 1 if there exists a directed transition from N_a to N_b, and 0 otherwise. This matrix is initialized using the sequence data, such that:

$$T_{a,b} = 1 \Leftrightarrow \exists i, j \mid (s_{i,j} = L_a^*) \wedge (s_{i,j+1} = L_b^*).$$

Once this baseline FSA is constructed, nodes can be split by looking for differences amongst groupings of exemplars, based on the label of the segment that preceded the exemplar in the observed execution; in other words, separating exemplars based on groupings of the node's parents in the FSA. For each node N_k with more than one parent, each possible split of the parents into two disjoint sets, P_{in} and P_{out} is

[1]The term 'node' is used rather than 'state' to avoid confusing it with the current observation or state of the robot.

examined, with associated exemplar sets E_{in} and E_{out}, and descendant sets D_{in} and D_{out} such that:

$$\begin{aligned}
E_{in} &:= \{e : e \in E_k \mid prev(e) \in P_{in}\} \\
E_{out} &:= \{e : e \in E_k \mid prev(e) \in P_{out}\} \\
D_{in} &:= \{next(e) : e \in E_{in}\} \\
D_{out} &:= \{next(e) : e \in E_{out}\}
\end{aligned}$$

Three criteria are then used to determine if the node should be split into two: (a) if the unique set of coordinate frame assignments corresponding to the exemplars in E_{in} is disjoint from that of E_{out}, (b) if the unique set of next segment labels (the $next(e)$) corresponding to the exemplars in E_{in} is disjoint from that of E_{out}, and (c) if the segment lengths of the exemplars in E_{in} come from a significantly different distribution than those in E_{out}. The difference in the length distributions is tested using the Kolmogorov-Smirnov test [58] as follows. Let F_{in} and F_{out} be the empirical cumulative distribution functions of the lengths of the exemplars in E_{in} and E_{out}. The test statistic z is then calculated based on the maximum distance between the empirical CDFs, without making any assumptions about the distributions involved:

$$\begin{aligned}
G &= \sup_x |F_{in}(x) - F_{out}(x)| \\
z &= G\sqrt{\frac{|E_{in}|\,|E_{out}|}{|E_{in}| + |E_{out}|}}.
\end{aligned}$$

This suggests the node should be split if $z < K_\alpha$, the critical value for significance value α. Here, $\alpha = 0.05$ is used.

If any of the three tests indicate to split the node, the node N_k is deleted and two new nodes N_{k-A} and N_{k-B} are created with exemplars E_{k-A} and E_{k-B}, such that:

$$T_{p,N_{k-A}} = 1 \Leftrightarrow p \in P_{in}$$

$$T_{p,N_{k-A}} = 1 \Leftrightarrow p \in P_{out}$$

$$T_{N_{k-B},d} = 1 \Leftrightarrow d \in D_{in}$$

$$T_{N_{k-B},d} = 1 \Leftrightarrow d \in D_{out}$$

$$E_{k-A} := E_{in}$$

$$E_{k-B} := E_{out}$$

This process, which we call *semantic splitting*, is then repeated, iterating over all the nodes until no more splits can be made.

Once the structure of the FSA is finalized, then a classifier is trained for each node that has multiple descendants. This is used as a transition classifier to determine which node to transition to next, once execution of a primitive at that node has taken place. For this task, a nearest neighbor classifier is used, but could be replaced by any multi-class classifier. For each classifier C_k corresponding to node N_k, training examples $Y_k = (y_{k,1}, \ldots, y_{k,r_k})$ are provided, corresponding to the final observation of the robot pose and object poses during each exemplar, plus a class label corresponding to the label of the node the given exemplar transitioned to next. Thus, training examples are recorded as $y \in SE(3)_1 \times \cdots \times SE(3)_{n+2} \times \mathbb{R}^2$, where n is the number of task objects. One exception to the above rule is the START node, which has no data associated with it; for this, the first observation of the first segment of the demonstration is used as training data.

Each classifier C_k is then trained using the training examples Y_k, along with their appropriate class labels. Once the classifier has been trained, future observations y can be classified so that appropriate transitions can be made at each decision point.

5.3.3 Task replay

Given a novel situation, the FSA can be used to replay the task by beginning at the START node that has no exemplars associated with it. The current observation is classified using C_{START} to determine which node to transition to first. This is a standard node N_i that contains exemplars E_i. To execute an appropriate movement, a DMP must be constructed from the exemplars; a single exemplar is chosen by examining the first observations associated with each exemplar and choosing the nearest candidate to the current observation y. The chosen exemplar is used to learn the weights for a DMP as described in Section 4.2.2.

To execute the DMP, the goal is shifted based on the coordinate frame that the segment was assigned to, so that the relative 6-DOF offset of the goal from the current origin of the coordinate frame is the same as it was in the exemplar. Then, the current robot pose is given as a starting state and the DMP is used to generate a movement plan to the goal with a resolution of 0.1 seconds. The plan is then executed, followed by a transition dictated by the node's transition classifier, using the current observation as input. This process is repeated until the END node is reached, a timeout occurs, or the robot is manually stopped by the user.

5.3.4 Interactive corrections

At any time during execution, the user can push a button on the joystick to stop the robot so that an interactive correction can be made. The robot immediately stops execution of the current movement and switches modes to accept a kinesthetic demonstration from the user. From the beginning of execution, the robot has been recording pose data in case of an interactive correction, and it continues to record as the user provides a demonstration of the remainder of the task.

After any number of replays and interactive corrections have taken place, the corrections can be integrated with the existing data for improved performance. All the old data plus the recordings from the interactive corrections are segmented from scratch; we do not begin with the old burned-in BP-AR-HMM, because the addition of new data may require a significantly different segmentation, and we wish to avoid systematic biasing of the sampling process. Following segmentation, the FSA is reconstructed as described above with the new data included, providing additional exemplars and refining the structure of the FSA. Algorithm 5 provides pseudocode for this iterative process.

5.4 Experiments

5.4.1 Experiment 1: demonstrations, corrections, and replay

We evaluated our system on a furniture assembly task, using a PR2 mobile manipulator to partially assemble a small off-the-shelf table[2]. The table consists of a tabletop with four pre-drilled holes and four legs that each have a screw protruding from one end. Eight kinesthetic demonstrations of the assembly task were provided, in which the tabletop and one leg were placed in front of the robot in various positions. In each demonstration, the robot was made to pick up the leg, insert the screw-end into the hole in the tabletop, switch arms to grasp the top of the leg, hold the tabletop in place, and screw in the leg until it is tight. An example of this progression is shown in Figure 5.2. To make the insertion more robust, a bit of human insight is applied by sliding the screw around the area of the hole until is slides in. This allows the insertion to be successful even when perception is slightly inaccurate.

The demonstrations were then segmented and and an FSA was built as described in Section 5.3. Figures 5.3(a)–(b) show the structure of the FSA before and after semantic splitting occurred; it can be seen that the node splits yield a much simpler, linear structure that better characterizes the flow of task progress, while leaving room

[2]Ikea LACK table: `http://www.ikea.com/us/en/catalog/products/20011413/`

Algorithm 5 Incremental Semantically Grounded Learning from Demonstration

```
Input: Demonstration trajectories $\mathbf{T} = (T_1, \ldots, T_N)$ and corresponding object pose
observations $\mathbf{O} = (O_1, \ldots, O_N)$

Loop as desired:

  1. Segment the demonstrations with the BP-AR-HMM:
       (segments, labels) = $Segmentation(\mathbf{T})$

  2. Find coordinate frame assignments:
       coord_frames = $CoordinateFrameDetection$(segments, labels, $\mathbf{O}$)

  3. Learn params of a DMP for each segment:
       dmps = $LearnDmpParams$(segments, coord_frames)

  4. Construct FSM:
       fsm = $BuildFSM$(segments)
       sfsm = $SemanticSplit$(fsm)
       classifiers = $TrainClassifiers$(sfsm, segments)

  5. Replay task based on current observations:
       state = START
       while state != END and $IsInterrupted$() == False do
          curr_obs = $GetCurrentObservation$()
          action = $Classify$(classifiers, state, curr_obs)
          state = $GetNextState$(sfsm, state, action)
          $ExecuteDmp$(action)
       end while

  6. Collect interactive correction if necessary:
       if $IsInterrupted$() = True then
          $(T_{corr}, O_{corr})$ = $GetCorrectionTrajectory$()
          $\mathbf{T} = (T_1, \ldots, T_N, T_{corr})$
          $\mathbf{O} = (O_1, \ldots, O_N, O_{corr})$
       end if
```

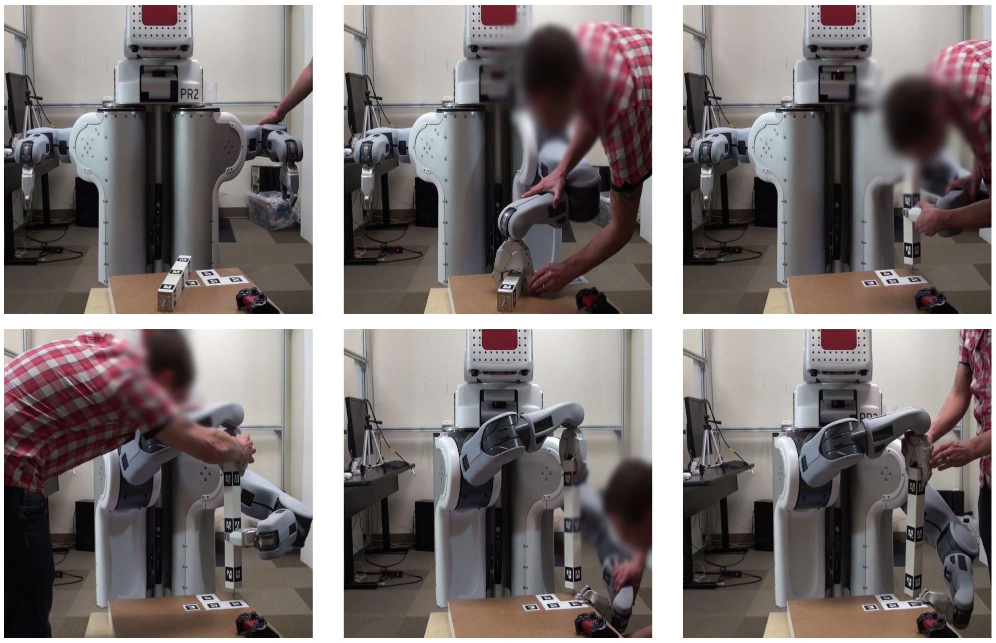
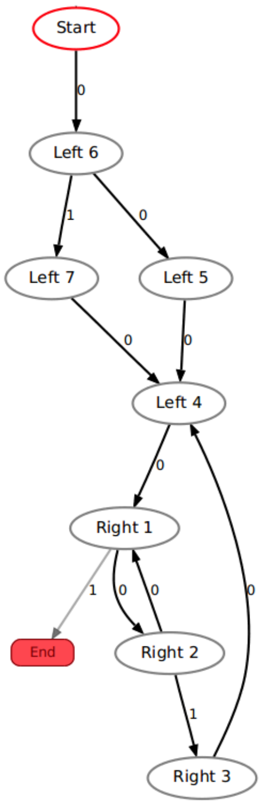


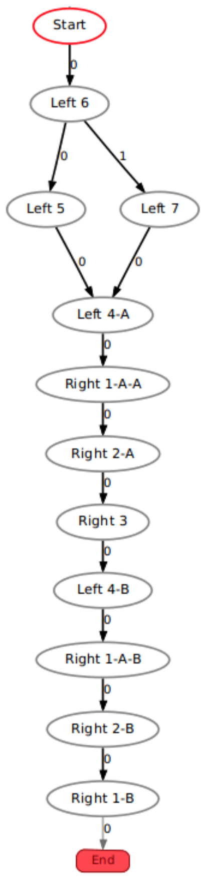
Figure 5.2. A kinesthetic demonstration of the table assembly task.

for branching when necessary. At this stage, task replay was sometimes successful, but several types of errors occurred intermittently. Two particular types of errors that occurred were (a) when the table leg was at certain angles, the robot was prone to missing the grasp, and (b) when the leg was too far from the robot, it could not reach far enough to grasp the leg at the desired point near the center of mass. In both cases interactive corrections were provided to recover from these contingencies. In the first case, a re-grasp was demonstrated, and then the task was continued as usual. In the second case, the robot was shown how to grasp the leg at a closer point, pull it towards itself, and then re-grasp it at the desired location.

After the interactive corrections were collected, the old data was re-segmented with the two new corrections and used to re-build the FSA. Figures 5.4(a)–(b) show the structure of the FSA before and after semantic splitting occurred. After splitting, this FSA is similar to that of Figure 5.3(b), but with several branching pathways near the beginning that then bottleneck into a linear flow. This is exactly the sort of structure that would be expected, given the nature of the corrections—the robot



(a) Demonstrations only, before semantic splitting

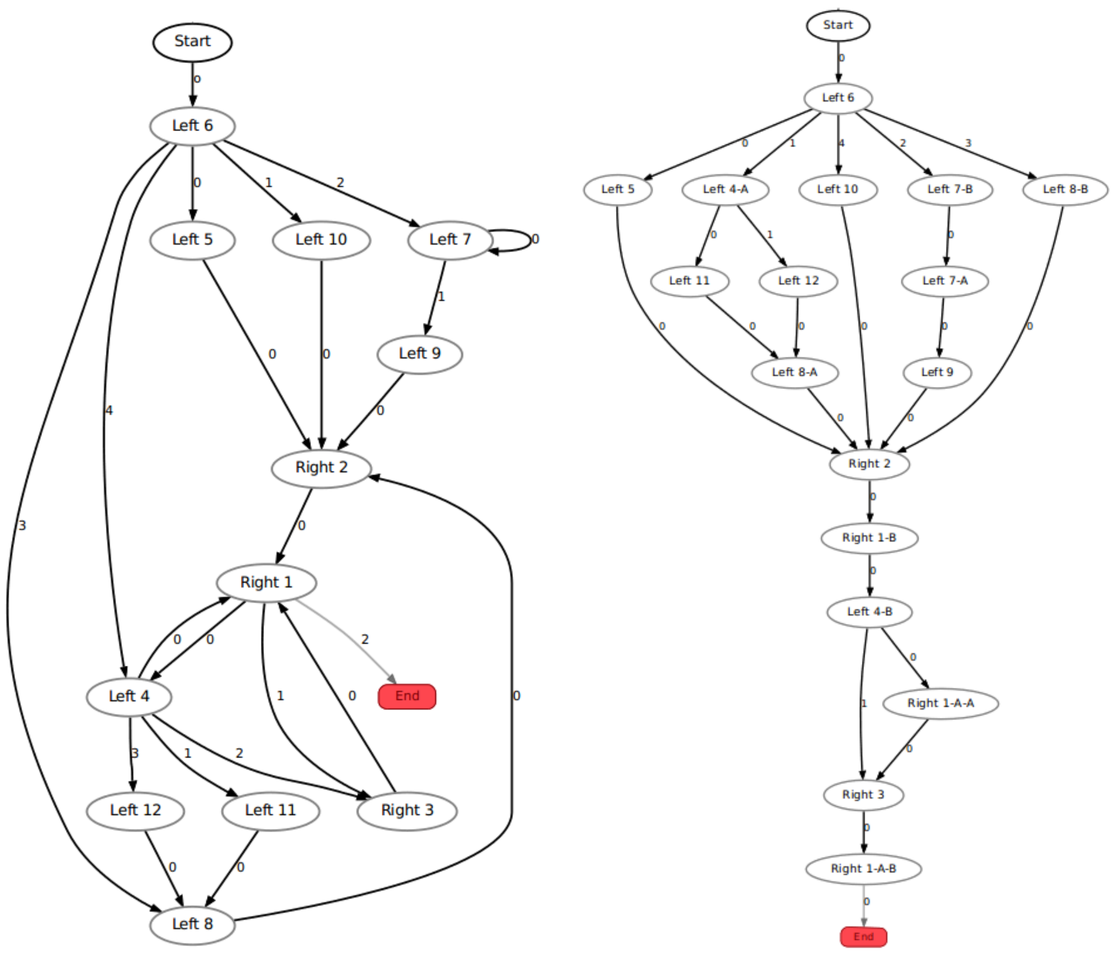


(b) Demonstrations only, after semantic splitting

Figure 5.3. FSA structure from demos before and after semantic splitting without interactive corrections. Nodes are labeled with the relevant arm (left/right), ID numbers, and A's and B's to denote splits.

attempts a grasp, chooses a path based on the outcome of the grasp, and then once the leg is successfully picked up and ready to be inserted, all paths converge on a common (nearly) linear sequence for the rest of the task.

Using this new FSA, the robot was able to recover from two types of errors in novel situations. Figure 5.5 shows a successful recovery from a missed grasp, while Figure 5.6 shows the robot bringing the table leg closer for a re-grasp when it is too far away. Finally, Figure 5.7 shows a full successful execution of the task without human



(a) Demonstrations + interactive corrections, before semantic splitting

(b) Demonstrations + interactive corrections, after semantic splitting

Figure 5.4. FSA structures before and after semantic splitting with interactive corrections added in. Nodes are labeled with the relevant arm (left/right), ID numbers, and A's and B's to denote splits.

intervention, demonstrating that these error recovery capabilities did not interfere with smooth execution in cases where no contingencies were encountered.

5.4.2 Experiment 2: method comparisons

The table in Figure 5.1 shows a quantitative comparison that demonstrates the benefits of FSA-based sequencing and interactive corrections. Ten trials of the table assembly task were attempted using four different sequencing methods with the segmented demonstration data from Experiment 1. The first method ('ASM') used

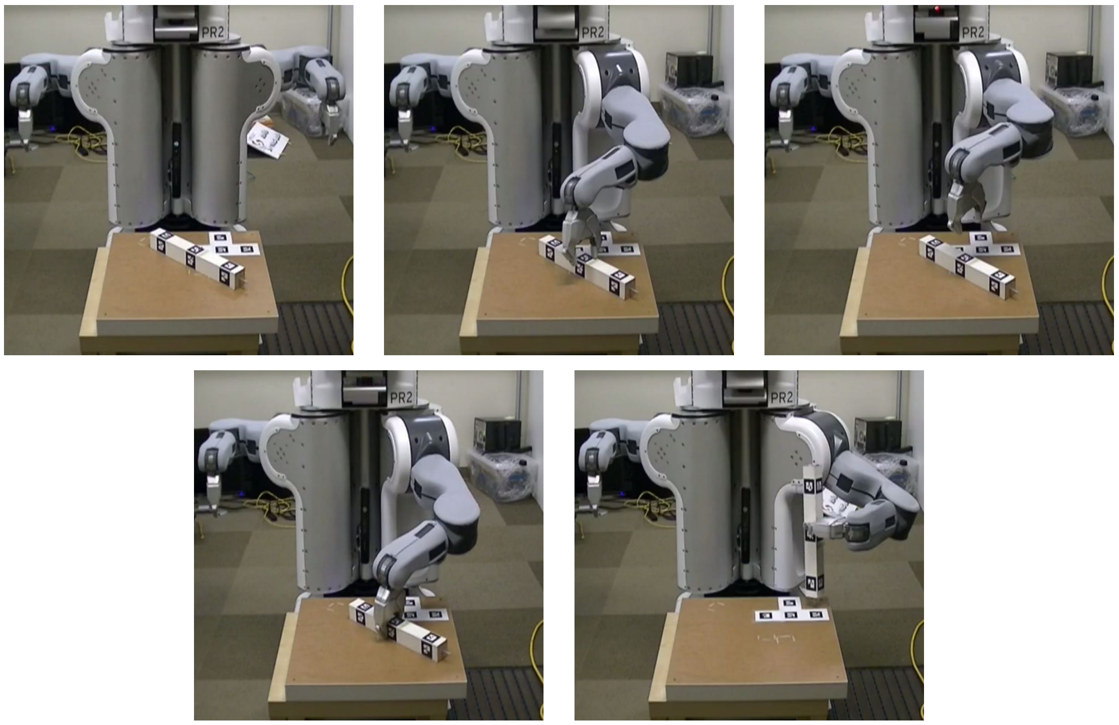


Figure 5.5. A recovery behavior when the robot misses the original grasp.

a framework similar to that of Associative Skill Memories by Pastor et al. [79], in which all primitives were available to be chosen at every decision point; classification was performed via a nearest neighbor classifier over the first observations of the exemplars associated with each movement category. The second ('FSA-basic') and third ('FSA-split') methods used an FSA before and after semantic splitting, respectively. Finally, the fourth method('FSA-IC') used an FSA after splitting with interactive corrections (also from Experiment 1) integrated as well.

Each set of ten trials was split up into three groups: four trials in which the table leg was straight and close to the PR2 ('Straight'), three trials in which the leg was too far away to grasp at the center of balance ('Far away'), and three trials in which the leg was at a difficult angle that could cause a missed grasp ('Difficult angle'). These were all novel configurations that had not been seen before, but the latter two were designed to produce situations similar to the interactive corrections collected earlier.

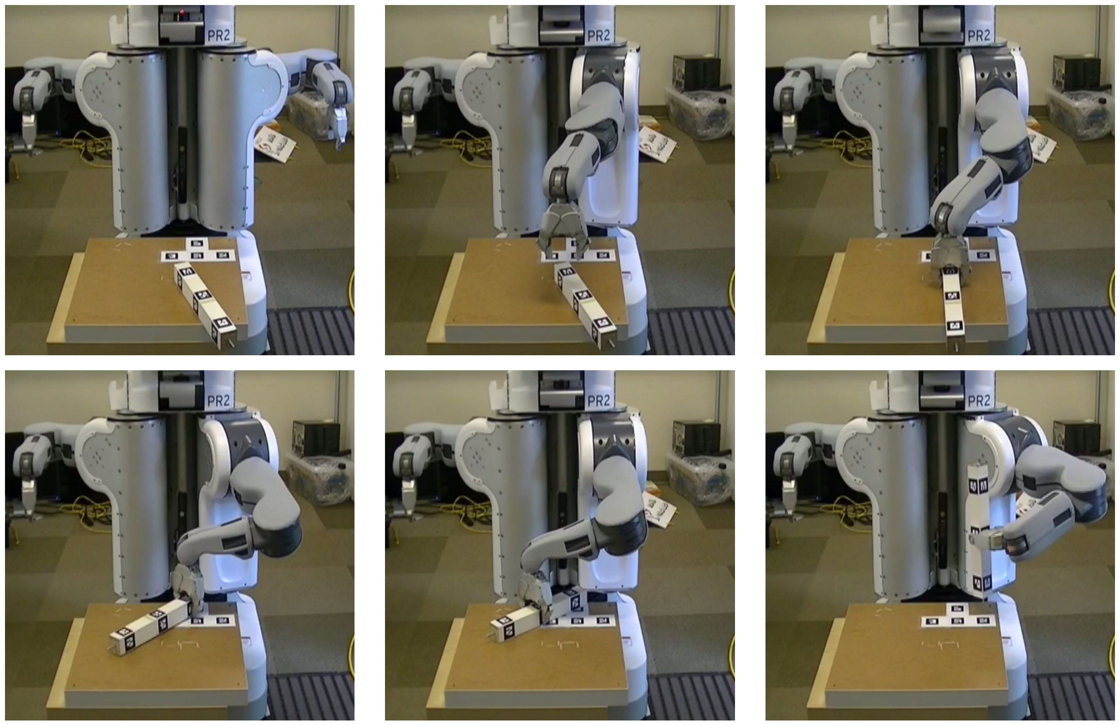


Figure 5.6. A recovery behavior when the table leg is too far away to grasp at the desired location.

During each trial, the operator was allowed to provide small assists to help the robot by moving an object or the robot's end effector by a maximum of 5 cm to compensate for minor perceptual or generalization errors. The entries in the table in Figure 5.1 denote the number of assists that were required during each trial, or 'Fail' if the robot failed to complete the task successfully. Here, we defined success as screwing the leg in far enough so that it was freestanding when the robot released it.

All ten trials with the 'ASM' and 'FSA-basic' methods resulted in failure, but for different reasons. While the ASM provided maximum sequencing flexibility, it also inherently made the classification problem difficult, since all choices were available at every step. Indeed, most failures were caused by misclassifications that caused the robot to choose inappropriate primitives or get stuck in an infinite loop, repeating the same primitive indefinitely. The FSA avoided infinite loops by reducing the number of choices at each decision point and providing ordering constraints between the nodes.

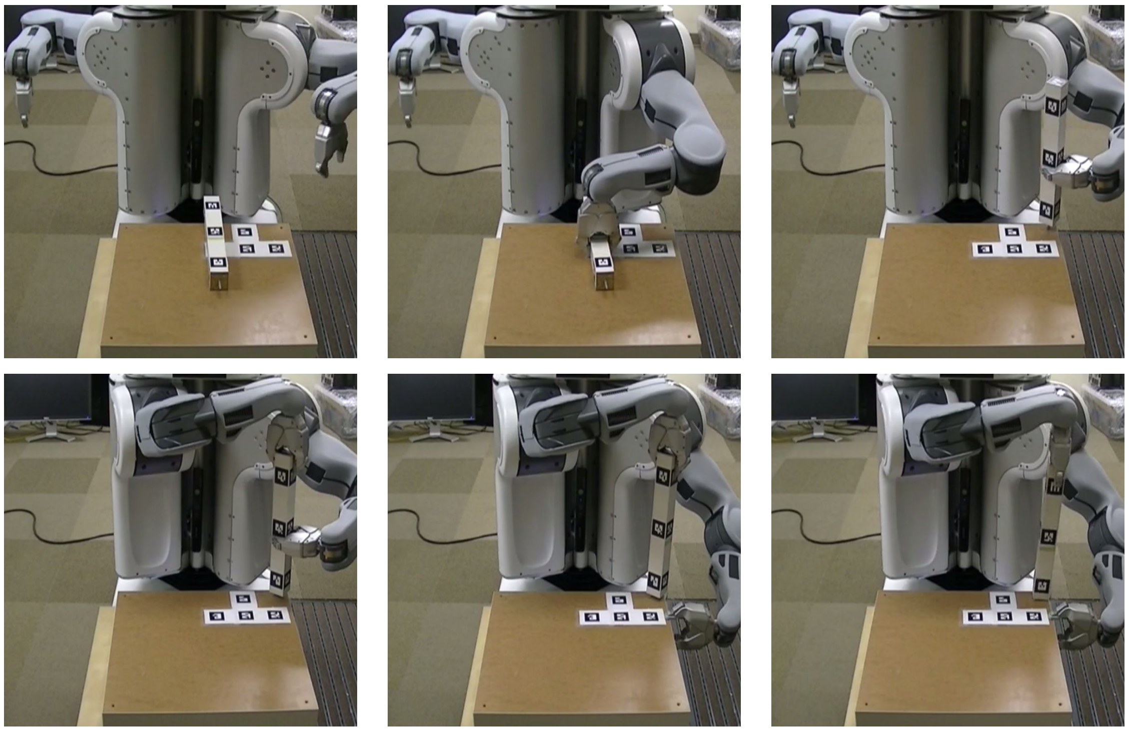


Figure 5.7. A full successful execution of the table assembly task without any human intervention. The task FSM is constructed using both the original demonstrations and the interactive corrections.

However, it still frequently chose poor exemplars or inappropriate primitives. Without semantic splitting, several semantically different primitives often got combined into a single node, corrupting the structure of the FSA, and making classification and exemplar selection more difficult.

Using semantic splitting, we observed seven successes and a modest number of required assists with the 'FSA-split' method, failing only in the 'Far away' case. In these cases, the robot reached for the leg until its arm was fully extended, but stopped far short of the intended grasp point; the difference was far too large to be fixed with a small assist. Once interactive corrections were added in 'FSA-IC', the number of successes increased to nine and the number of required assists dropped by more than 25%—a significant improvement. The remaining assists were largely due to small perceptual errors during the screw insertion, which requires a high level of precision.

	ASM	FSA-basic	FSA-split	FSA-IC
Straight	Fail	Fail	1	0
	Fail	Fail	1	2
	Fail	Fail	2	2
	Fail	Fail	1	2
Far away	Fail	Fail	Fail	1
	Fail	Fail	Fail	1
	Fail	Fail	Fail	Fail
Difficult angle	Fail	Fail	2	1
	Fail	Fail	3	1
	Fail	Fail	3	2
Successes / Avg assists	0 / –	0 / –	7 / 1.857	9 / 1.333

Table 5.1. Ten trials of the task with corresponding performance data for four different types of sequencing: an associative skill memory (ASM), an FSA without semantic splitting (FSA-basic), an FSA with semantic splitting (FSA-split), and an FSA with splitting and interactive corrections (FSA-IC). 'Fail' indicates failure and integer values correspond to the number of human assists required during successful executions.

These errors appeared to be random, lacking structure that could be leveraged; additional interactive corrections were provided, but did not further improve performance. This reveals an important limitation of interactive corrections—in general, they can only be used to recover from observable, structured errors.

5.4.3 Experiment 3: looping behaviors

For this final experiment we designed a task that contains a behavioral loop, with the goal of capturing this loopy structure in the FSA. Figure 5.8 shows a demonstration of a peg-in-hole task in which the robot must pick up a rectangular peg and insert it in the hole in the center of the table. In three of the demonstrations of this task, the robot was shown how to complete the task normally by picking up the peg and inserting it in the hole immediately. However, in eight other demonstrations, trajectories were provided that slightly miss the hole several times before successfully

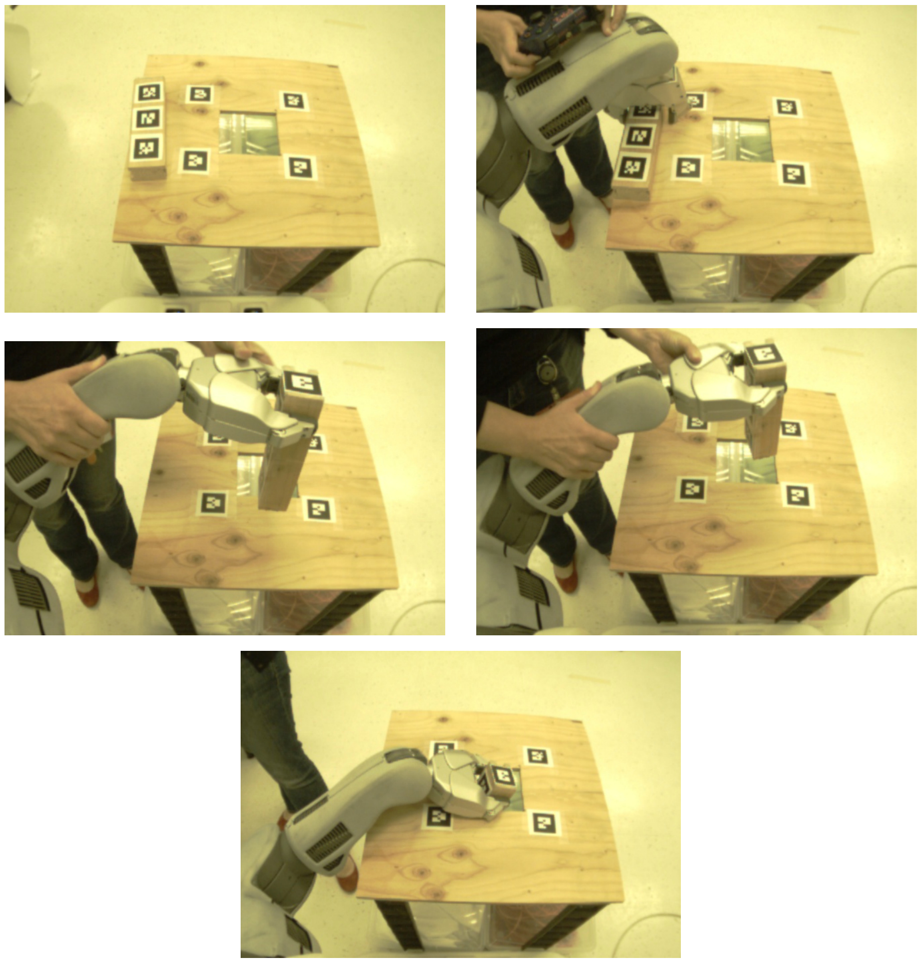


Figure 5.8. A demonstration of the peg-in-hole task. Panels 3 and 4 show failed insertions attempts and and retries, while panel 5 shows the final successful insertion.

getting the peg in. This is a looping "retry" behavior, in which the peg is lifted in the air and attempted to be inserted again until successful.

The goal of this experiment was for the robot to learn a loopy FSA, such that it would automatically retry the peg insertion step until it was classified as successful. For such an FSA to be learned, the segmentation of the demonstrations would have to identify each insertion attempt as a distinct segment. Unfortunately, this was not the case; Figure 5.9 shows a segmentation of the demonstrations in which all insertion attempts are clumped together into one long segment at the end of each demo (the green segment). This final segment is longer or shorter in each demo depending on the number of repetitions, but is never split into more than one segment.

This failure reveals an interesting feature and limitation of our algorithm. Our method makes no distinction between seeing the same motion category several times in succession and seeing that motion category once for a longer period of time—each segment is just a contiguous string of observations where the same motion category is active, so there cannot be multiple adjacent segments of identical motion categories. Thus, since the system identified the lifting and insertion of the peg all as a single motion category, it was not possible to segment out each repetition of that motion. Interestingly, if lifting the peg and inserting the peg were identified as separate motion categories, the repeating pattern would have been able to be discovered as a two-segment cycle.

Thus, we conclude that in principle, the proposed system can find looping structure in tasks, but in practice it may often fail, due to the granularity at which segmentation occurs. At a deeper level, this may be a fundamental problem with a segmentation system that infers statistical motion categories rather than subgoals that the demonstrations imply. Future work may examine ways to overcome this difficulty.

5.5 Related Work

The most closely related work to ours is that of Grollman et al. [39], which addresses *perceptual aliasing* by using a nonparametric Bayesian model to infer a mixture of experts from unsegmented demonstration data and then using multi-map regression to assign observed states to experts. Butterfield et al. [17] extend this work by getting rid of the assumption that data is independent and identically distributed, leading to policies that better model the time-series nature of tasks.

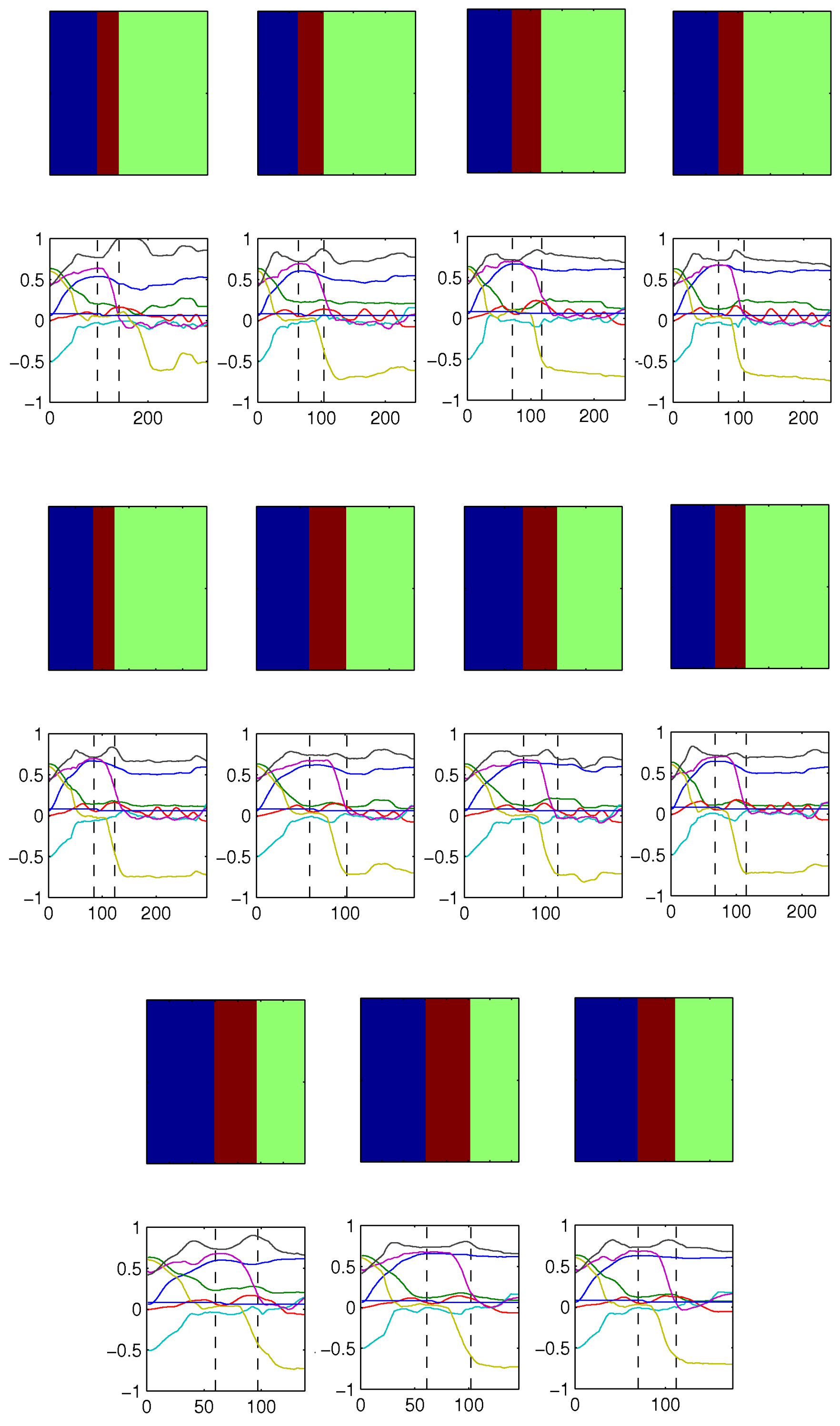


Figure 5.9. Segmentation of 8 demonstrations of the peg-in-hole task with repeated retries of insertions (top 2 rows) and 3 demonstrations of successful first-try insertions (bottom row).

Other work aims to intelligently sequence predefined primitives, rather than discover them from data. Toussaint et al. [104] use heuristics to translate perceptions into predefined symbols, which are then used by a rule-based planner. In the control basis framework [44], a graph of admissible behaviors is automatically built based on the predicates and constraints of multiple hand-crafted controllers, allowing safe composite policies to be learned. Given a set of predefined skills, Riano et al. [87] evolve the structure of a finite-state automaton that defines transition behavior between the skills. Pastor et al. [79] demonstrate one skill at a time and then use nearest neighbor classification to associate skills with observations, while also monitoring skill executions for proprioceptive anomalies so that recovery behaviors can be initiated. Wilcox et al. [106] use a dynamic scheduling algorithm to adapt execution of a predefined set of events to a user's preferences and temporal disruptions, while maintaining critical synchronization invariants. Ekvall and Kragic [28] search for skill ordering constraints in tasks with many valid execution orderings.

Several other approaches also allow users to interactively provide additional data and corrections to a robot. Thomaz and Breazeal [102] provide an interface that allows users to bias the robot's exploration and provide positive and negative feedback during task execution. Another approach uses user-defined keyframes that identify critical points in a task that must be reproduced [2]. Upon replay, the teacher can use an interface to view and edit keyframes to fine-tune task performance. Muhlig et al. [63] rely entirely on real-time human-robot interaction to ensure proper sequencing of skills and recovery from errors, using hand signals and speech to control skill flow, halt execution, and identify important task objects.

5.6 Conclusion

Flexible discovery and sequencing of primitives is essential for tractable learning of complex robotic tasks from demonstration. We introduced a novel method to split automatically discovered motion categories into semantically grounded primitives, sequence them intelligently, and provide interactive corrections for incremental performance improvement. Sequencing primitives with a finite-state automaton allows exemplars of movement primitives to be grouped together in a semantically meaningful way that attempts to maximize data reuse, while minimizing the number of options that the agent must choose amongst at each step. This approach makes the sequencing classification task easier, while also providing a mechanism for semantically grounding each primitive based on state visitation history and observed characteristics like coordinate frame, length, and successor state.

This approach was validated on a furniture assembly task using a PR2 mobile manipulator. It was shown that the robot could learn the basic structure of the task from a small number of demonstrations, which were supplemented with interactive corrections as the robot encountered contingencies that would have lead to failure. The corrections were then used to refine the structure of the FSA, leading to new recovery behaviors when these contingencies were encountered again, without disrupting performance in the nominal case. A quantitative measure was also provided to show that using an FSA with semantic splitting provides advantages that lead to improved classification and task performance compared to more naive methods.

While performance was able to be improved through interactive corrections, future work could include a mechanism to improve task performance and individual primitives automatically though self-practice. Additionally, we only took advantage of the multiple exemplars of each primitive by selecting amongst them; in the future, it would be beneficial to integrate the exemplars to better model the user's intentions. Only vision and pose data were used as part of the discriminative state space, but several other types of input such as force data could be valuable for decision making, or even for modulating our DMPs, as in [80]. Finally, the useful looped structure of repetitive behavior proved difficult to capture, so future work may look at ways to capture contiguous repetitions of the same movement. With improvements such as these, the presented method might function as one element of a deployable system that allows researchers and end users alike to efficiently program robots via unsegmented, natural demonstrations.

CHAPTER 6
DISCUSSION AND CONCLUSION

The goal of this thesis was to develop techniques that extend the generalization capabilities of learning from demonstration algorithms in complex robotic tasks. This goal was met through the development of three novel skill discovery algorithms, culminating in an incremental method for learning semantically meaningful skills from demonstration. First, we introduced a nonparametric Bayesian clustering method that can automatically learn an appropriate number of skills to accomplish important subtasks in a simulated reinforcement learning setting. Next, a tractable algorithm for use on physical robots was introduced that leverages additional data from task demonstrations. In this method, repeated structure was discovered through nonparametric Bayesian segmentation and coordinate frame detection algorithms to create flexible skill controllers. Finally, an algorithm was introduced to incrementally learn and improve a finite-state representation of the task that allows for intelligent, adaptive sequencing of skills and recovery from errors. Skills in the FSA were correlated with external observations, imbuing the skills with semantic meaning and greater reusability.

Several experiments were performed to validate these algorithms, both in simulation and on a PR2 mobile manipulator. We showed how our methods can be used to teach robots complex, multi-step tasks from a relatively small number of demonstrations. In the final set of experiments in Chapter 5, the PR2 was able to learn to partially assemble a small IKEA table from only 8 demonstrations. Performance was then improved by providing interactive corrections to the robot as mistakes were made during execution. Finally, it was shown that this combination of techniques allowed the robot to incrementally learn to perform a task, recover from errors, and adapt to novel situations and unforeseen contingencies.

6.1 Future Work

The techniques presented in this thesis reveal new possibilities for more generally applicable robot learning from demonstration algorithms. However, both the presented algorithms and experiments are necessarily limited in scope, leaving many opportunities for improvement in future work.

A key piece of our system, the Beta Process Autoregressive Hidden Markov Model, may be able to be improved in several ways that could lead to improved segmentation and performance. We used the BP-AR-HMM to find repeated dynamical systems in the data and then performed coordinate frame detection as a separate step afterwards. Ideally, this would be done all as a single step, in a principled, fully Bayesian way during inference on the model. Aside from being more principled, performing coordinate frame detection within the model might allow us to find repeated dynamical systems that would have previously eluded the BP-AR-HMM; some motions may look very different in the robot frame, but may look very similar in some other object's frame of reference. Additionally, the development of more efficient inference mechanisms would allow our method to work on larger data sets. Finally, it may be worthwhile to explore other types of time-series dynamics than linear dynamical systems.

Our system only considered spatial data for the purposes of evaluating the state of the world and creating DMP plans. However, other sensing modalities such as force/torque could be used to enhance the capabilities of the robot. Force sensing, for example, could allow the robot to feel whether a screw was in place, rather than having to guess based on noisy and/or occluded spatial observations. Some work has already been done by Pastor et al. [80] on adapting DMP plans on-line based on force data so that executions “feel” the same to the robot as they did during demonstrations, resulting in improved performance and robustness to small errors.

However, complex concepts like “screw is in hole” may require an additional level of semantic understanding that has not yet been reached with current techniques. Our approach gave semantic meaning to states in a finite-state representation of a task by correlating each state with observations of the world or associated actions executed when in that state. However, more abstract, complex concepts may require *functional analysis*—for example, realizing that both a fork and a toothpick can be used to pick up a piece of cheese, since they both have pointed ends to skewer it with.

Task replay can also be improved through further innovations. Rather than integrate all examples of a particular skill that we have identified, our system uses a simple nearest-neighbor strategy to choose the single most relevant example to use for DMP training. However, all the available data could be used more efficiently if a principled way to integrate this data was designed. However, it is well known that the average of solutions to a problem is often not itself a solution, making this problem non-trivial, though some approaches have been explored [22, 13]. Furthermore, by using DMPs, we are performing somewhat “blind” replay of skills. If preconditions and postconditions for each skill could be discovered, more traditional planning algorithms could be used to perform each skill more adaptively. Finally, for more dynamic tasks than we considered in this thesis, each skill could also be improved through reinforcement learning if necessary, by inferring a reward function through inverse reinforcement learning.

The experiment in this thesis were limited to a single task at a time with relatively few demonstrations. However, in the future, algorithms such as these must scale to demonstrations of many tasks, producing a library of skills over time that can be reused in many situations. This requires more efficient inference algorithms, as well as strategies to manage such a library of skills and to choose appropriate skills in various situations. Furthermore, collecting all this data of many different tasks may take more time than any one user is willing to put in. Further research into web robotics and remote laboratories [76] may provide mechanisms to crowdsource this type of data collection.

Finally, natural language integration may be useful for providing a simpler interface for the user to command tasks and give feedback to the robot. This could allow the user to teach the robot the names of objects, actions, and tasks during demonstrations. The robot may be able to infer associations between language and actions, allowing it to perform novel tasks from natural language commands, as long as it has seen similar components of the task before. In fact, some existing related work has already demonstrated a principled system for inferring user intentions from natural language [100]. Natural language commands could also be used for feedback on a task, such as “good” and “bad”, or to modify the robot’s behavior, like “slower” or “harder”.

6.2 Conclusion

A perennial goal of robotics has been the creation of robots that can exhibit a wide range of intelligent behaviors in diverse environments. While advances in machine learning techniques have improved the quality of such learned behaviors, both researchers and end-users alike need tools that can help them deal with the vast number of behaviors and environments that must be mastered by a robot deployed in the real world. Learning from demonstration has shown to be a promising paradigm to quickly program robots in a natural way, but has often been limited by poor generalization.

We have shown that learning from demonstration algorithms can partially overcome their traditionally limited generalization capabilities by acquiring semantically meaningful skills through the discovery of repeated statistical structure in demonstrations. The discovery of repeated structure provided critical insights into task invariants, features of importance, high-level task structure, and appropriate skills for the task. This allowed the robot to reproduce complex, multi-step tasks in novel situations by learning from a small number of unstructured, whole-task demonstrations and interactive corrections.

To achieve this goal, this thesis drew from recent advances in Bayesian nonparametric statistics and control theory to develop novel learning from demonstration algorithms and an integrated system capable of learning tasks from start to finish; this system combined perception, kinesthetic input, control, segmentation, clustering, and classification algorithms to enable experimentation on a PR2 mobile manipulator. A series of experiments were conducted, culminating in a complex table assembly task that the PR2 was able to reproduce in a number of novel situations, even when contingencies and error conditions were encountered.

While this thesis advanced the state of the art in learning from demonstration algorithms, we also believe that the integrated system itself is an equally important contribution. As robotics attempts to move forward into the home and workplace, it is our belief that many of the most difficult technical problems will be integrative in nature, rather than limited to a particular subfield such as perception or control. The scope of our work was necessarily limited and leaves much room for future improvement, but we have provided the basis for a principled way identify and leverage semantic structure in demonstration data. Going forward, we hope this can serve as a step toward a deployable system that allows researchers and end users alike to efficiently program robots, paving the way for robots in the home and workplace.

BIBLIOGRAPHY

[1] Abbeel, Pieter, and Ng, Andrew Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty First International Conference on Machine Learning* (2004), pp. 1–8.

[2] Akgun, Baris, Cakmak, Maya, Yoo, Jae Wook, and Thomaz, Andrea Lockerd. Trajectories and keyframes for kinesthetic teaching: a human-robot interaction perspective. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction* (2012), pp. 391–398.

[3] Aldous, D.J., Ibragimov, I.A., and Jacob, J. *Exchangability and Related Topics.* Lecture notes in mathematics. Springer-Verlag, 1983.

[4] Argall, B.D., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robotics and Autonomous Systems 57*, 5 (2009), 469–483.

[5] Atkeson, C. G., and Schaal, S. Robot learning from demonstration. In *Proceedings of the Fourteenth International Conference on Machine Learning* (1997), pp. 12–20.

[6] Bakker, Bram, and Schmidhuber, Jürgen. Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization. In *Proceedings of the 8th Conference on Intelligent Autonomous Systems* (2004), pp. 438–445.

[7] Barto, A. G., Singh, S., and Chentanez, N. Intrinsically motivated learning of hierarchical collections of skills. In *Proc. of the International Conference on Developmental Learning* (2004), pp. 112–119.

[8] Bauml, Berthold, Wimbock, T, and Hirzinger, Gerd. Kinematically optimal catching a flying ball with a hand-arm-system. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2010), pp. 2592–2599.

[9] Beal, Matthew J., Ghahramani, Zoubin, and Rasmussen, Carl E. The infinite hidden markov model. In *Machine Learning* (2002), MIT Press, pp. 29–245.

[10] Beaumont, M.A., and Rannala, B. The bayesian revolution in genetics. *Nature Reviews Genetics 5*, 4 (2004), 251–61.

[11] Billard, A., Calinon, S., Dillmann, R., and Schaal, S. *Handbook of Robotics.* Springer, Secaucus, NJ, USA, 2008, ch. Robot programming by demonstration.

[12] Bishop, Christopher M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007.

[13] Bitzer, Sebastian, and Vijayakumar, Sethu. Latent spaces for dynamic movement primitives. In *9th IEEE-RAS International Conference on Humanoids* (2009), pp. 574–581.

[14] Blei, David M., Ng, Andrew Y., and Jordan, Michael I. Latent dirichlet allocation. *Journal of Machine Learning Research 3* (2003), 993–1022.

[15] Bollini, Mario, Barry, Jennifer, and Rus, Daniela. Bakebot: Baking cookies with the pr2. In *The PR2 Workshop: Results, Challenges and Lessons Learned in Advancing Robots with a Common Platform, IROS* (2011).

[16] Boularias, A., Kober, J., and Peters, J. Relative entropy inverse reinforcement learning. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling* (2011), pp. 20–27.

[17] Butterfield, J., Osentoski, S., Jay, G., and Jenkins, O.C. Learning from demonstration using a multi-valued function regressor for time-series data. In *Proceedings of the Tenth IEEE-RAS International Conference on Humanoid Robots* (2010).

[18] Calinon, S., and Billard, A. Incremental learning of gestures by imitation in a humanoid robot. In *Proceedings of the Second Conference on Human-Robot Interaction* (2007).

[19] Cederborg, Thomas, Li, Ming, Baranes, Adrien, and Oudeyer, P-Y. Incremental local online gaussian mixture regression for imitation learning of multiple tasks. In *IEEE/RSJ International Conference onIntelligent Robots and Systems* (2010), pp. 267–274.

[20] Chernova, Sonia, and Veloso, Manuela. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems* (2007).

[21] Chiappa, S., and Peters, J. Movement extraction by detecting dynamics switches and repetitions. *Advances in Neural Information Processing Systems 23* (2010), 388–396.

[22] Coates, A., Abbeel, P., and Ng, A.Y. Learning for control from multiple demonstrations. In *Proceedings of the 25th International Conference on Machine Learning* (2008), ACM, pp. 144–151.

[23] de Finetti, B. *Funzione Caratteristica Di un Fenomeno Aleatorio*. 6. Memorie. Academia Nazionale del Linceo, 1931, pp. 251–299.

[24] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B 39*, 1 (1977), 1–38.

[25] Digney, Bruce L. Learning hierarchical control structures for multiple tasks and changing environments. In *Proceedings of the 5th Conference on the Simulation of Adaptive Behavior* (1998), MIT Press.

[26] Dixon, K.R., and Khosla, P.K. Trajectory representation using sequenced linear dynamical systems. In *IEEE International Conference on Robotics and Automation* (2004), vol. 4, IEEE, pp. 3925–3930.

[27] Dong, S., and Williams, B. Motion learning in variable environments using probabilistic flow tubes. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on* (2011), IEEE, pp. 1976–1981.

[28] Ekvall, S., and Kragic, D. Learning task models from multiple human demonstrations. In *15th IEEE International Symposium on Robot and Human Interactive Communication* (2006), pp. 358–363.

[29] Ekvall, Staffan, and Kragic, Danica. Robot learning from demonstration: a task-level planning approach. *International Journal of Advanced Robotic Systems 5*, 3 (2008), 223–234.

[30] Fox, E.B. *Bayesian Nonparametric Learning of Complex Dynamical Phenomena.* Ph.D. thesis, MIT, Cambridge, MA, 2009.

[31] Fox, E.B., Sudderth, E.B., Jordan, M.I., and Willsky, A.S. An HDP-HMM for systems with state persistence. In *Proceedings of the 25th International Conference on Machine Learning* (2008), ACM, pp. 312–319.

[32] Fox, E.B., Sudderth, E.B., Jordan, M.I., and Willsky, A.S. Sharing features among dynamical systems with beta processes. *Advances in Neural Information Processing Systems 22* (2009), 549–557.

[33] Fox, E.B., Sudderth, E.B., Jordan, M.I., and Willsky, A.S. Joint modeling of multiple related time series via the beta process. *arXiv:1111.4226* (2011).

[34] Geman, Stuart, and Geman, Donald. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 6*, 6 (1984), 721–741.

[35] Gienger, M., Muhlig, M., and Steil, J.J. Imitating object movement skills with robots: A task-level approach exploiting generalization and invariance. In *International Conference on Intelligent Robots and Systems* (2010), IEEE, pp. 1262–1269.

[36] Gilks, W. R., and Wild, P. Adaptive Rejection Sampling for Gibbs Sampling. *Journal of the Royal Statistical Society, Series C 41*, 2 (1992), 337–348.

[37] Griffiths, Thomas L., and Ghahramani, Zoubin. The indian buffet process: An introduction and review. *Journal of Machine Learning Research 12* (2011), 1185–1224.

[38] Grollman, D. H., and Jenkins, O. C. Sparse incremental learning for interactive robot control policy estimation. In *Proceedings of the International Conference on Robotics and Automation* (2008).

[39] Grollman, D H, and Jenkins, O C. Incremental learning of subtasks from unsegmented demonstration. *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2010), 261–266.

[40] Grollman, D.H., and Jenkins, O.C. Incremental learning of subtasks from unsegmented demonstration. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2010), IEEE, pp. 261–266.

[41] Halkidi, M., and Vazirgiannis, M. Npclu: An approach for clustering spatially extended objects. *Intell. Data Anal. 12* (December 2008), 587–606.

[42] Hastings, W.K. Monte carlo samping methods using markov chains and their applications. *Biometrika* (1970), 97–109.

[43] Hewitt, Edwin, and Savage, Leonard J. Symmetric measures on cartesian products. *Transactions of the American Mathematical Society 80*, 2 (1955), 470–501.

[44] Huber, Manfred, and Grupen, Roderic A. Learning to coordinate controllers - reinforcement learning on a control basis. In *Proceedings of the International Joint Conference on Artificial Intelligence* (1997), pp. 1366–1371.

[45] Ijspeert, A.J., Nakanishi, J., and Schaal, S. Learning attractor landscapes for learning motor primitives. *Advances in Neural Information Processing Systems 16* (2003), 1547–1554.

[46] Ipek, Engin, Mutlu, Onur, Martinez, Jose F., and Caruana, Rich. Self-optimizing memory controllers: A reinforcement learning approach. *Computer Architecture, International Symposium on 0* (2008), 39–50.

[47] Jenkins, Odest Chadwicke, and Matarić, Maja. A spatio-temporal extension to Isomap nonlinear dimension reduction. In *Proceedings of the Twenty-First International Conference on Machine Learning* (Jul 2004), pp. 441–448.

[48] Jenkins, Odest Chadwicke, and Matarić, Maja J. Performance-derived behavior vocabularies: Data-driven acquisition of skills from motion. *International Journal of Humanoid Robotics 1*, 2 (Jun 2004), 237–288.

[49] Jonsson, Anders, and Barto, Andrew. Causal graph based decomposition of factored mdps. *J. Mach. Learn. Res. 7* (December 2006), 2259–2301.

[50] Kjellström, Hedvig, Romero, Javier, and Kragić, Danica. Visual object-action recognition: Inferring object affordances from human demonstration. *Computer Vision and Image Understanding 115*, 1 (2011), 81–90.

[51] Konidaris, G., Kuindersma, S., Grupen, R., and Barto, A. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research 31*, 3 (2012), 360–375.

[52] Konidaris, G. D., Kuindersma, S. R., Barto, A. G., and Grupen, R. A. Constructing skill trees for reinforcement learning agents from demonstration trajectories. In *Advances in Neural Information Processing Systems 23 (NIPS 2010)* (2010).

[53] Konidaris, G.D., Osentoski, S., and Thomas, P.S. Value function approximation in reinforcement learning using the Fourier basis. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence* (2011).

[54] Konidaris, George, and Barto, Andrew G. Building portable options: Skill transfer in reinforcement learning. In *Proc. of the 20th International Joint Conference on Artificial Intelligence* (2007), pp. 895–900.

[55] Konidaris, George, and Barto, Andrew G. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems 22* (2009), pp. 1015–1023.

[56] Kulic, D., Takano, W., and Nakamura, Y. Online segmentation and clustering from continuous observation of whole body motions. *IEEE Transactions on Robotics 25*, 5 (2009), 1158–1166.

[57] Lucas, Peter J F, van der Gaag, Linda C, and Abu-Hanna, Ameen. Bayesian networks in biomedicine and health-care. *Artificial intelligence in medicine 30*, 3 (2004), 201–14.

[58] Massey, Frank J Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American Statistical Association 46*, 253 (1951), 68–78.

[59] McGovern, Amy, and Barto, Andrew G. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the 18th International Conference on Machine Learning* (2001), pp. 361–368.

[60] Metropolis, Nicholas, Rosenbluth, Arianna W., Rosenbluth, Marshall N., Teller, Augusta H., and Teller, Edward. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics 21*, 6 (1953), 1087–1092.

[61] Miller, Stephen, Van Den Berg, Jur, Fritz, Mario, Darrell, Trevor, Goldberg, Ken, and Abbeel, Pieter. A geometric approach to robotic laundry folding. *The International Journal of Robotics Research 31*, 2 (2012), 249–267.

[62] Moore, Brett, Panousis, Periklis, Kulkarni, Vivek, Pyeatt, Larry, and Doufas, Anthony. Reinforcement learning for closed-loop propofol anesthesia: A human volunteer study. In *Innovative Applications of Artificial Intelligence* (2010).

[63] Mühlig, Manuel, Gienger, Michael, and Steil, Jochen J. Interactive imitation learning of object movement skills. *Autonomous Robots 32*, 2 (2011), 97–114.

[64] Nakanishi, Jun, Morimoto, Jun, Endo, Gen, Cheng, Gordon, Schaal, Stefan, and Kawato, Mitsuo. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems 47*, 2 (2004), 79–91.

[65] Navarro, Daniel J., Griffiths, Thomas L., Steyvers, Mark, and Lee, Michael D. Modeling individual differences using dirichlet processes. *Journal of Mathematical Psychology 50*, 2 (2006), 101–122.

[66] Neal, R.M. Markov chain sampling methods for Dirichlet process mixture models. *Journal of computational and graphical statistics 9*, 2 (2000), 249–265.

[67] Neu, G., and Szepesvári, C. Apprenticeship learning using inverse reinforcement learning and gradient methods. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence* (2007), pp. 295–302.

[68] Ng, A., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., and Liang, E. Autonomous inverted helicopter flight via reinforcement learning. *Experimental Robotics IX* (2006), 363–372.

[69] Ng, Andrew Y., Jordan, Michael I., and Weiss, Yair. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems* (2001), MIT Press, pp. 849–856.

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

[71] Nicolescu, M., and Matarić, M. J. Natural methods for robot task learning: Instructive demonstration, generalization and practice. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems* (2003), pp. 241–248.

[72] Niekum, S., Barto, A.G., and Spector, L. Genetic programming for reward function search. *IEEE Transactions on Autonomous Mental Development 2*, 2 (2010), 83–90.

[73] Niekum, Scott, and Barto, Andrew G. Clustering via dirichlet process mixture models for portable skill discovery. In *Advances in Neural Information Processing Systems 24* (2011), pp. 1818–1826.

[74] Niekum, Scott, Chitta, Sachin, Marthi, Bhaskara, Osentoski, Sarah, and Barto, Andrew G. Incremental semantically grounded learning from demonstration. In *Robotics: Science and Systems* (2013).

[75] Niekum, Scott, Osentoski, Sarah, Konidaris, George, and Barto, Andrew G. Learning and generalization of complex tasks from unstructured demonstrations. *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2012), 5239–5246.

[76] Osentoski, Sarah, Pitzer, Benjamin, Crick, Christopher, Jay, Graylin, Dong, Shuonan, Grollman, Daniel, Suay, Halit Bener, and Jenkins, Odest Chadwicke. Remote robotic laboratories for learning from demonstration. *International Journal of Social Robotics 4*, 4 (2012), 449–461.

[77] Pastor, P., Hoffmann, H., Asfour, T., and Schaal, S. Learning and generalization of motor skills by learning from demonstration. In *IEEE International Conference on Robotics and Automation* (2009), IEEE, pp. 763–768.

[78] Pastor, Peter, Kalakrishnan, Mrinal, Chitta, Sachin, Theodorou, Evangelos, and Schaal, Stefan. Skill learning and task outcome prediction for manipulation. In *Proceedings of the 2011 IEEE International Conference on Robotics & Automation* (2011).

[79] Pastor, Peter, Kalakrishnan, Mrinal, Righetti, Ludovic, and Schaal, Stefan. Towards associative skill memories. *IEEE-RAS International Conference on Humanoid Robots* (2012).

[80] Pastor, Peter, Righetti, Ludovic, Kalakrishnan, Mrinal, and Schaal, Stefan. Online movement adaptation based on previous sensor experiences. *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2011), 365–371.

[81] Peters, J., Vijayakumar, S., and Schaal, S. Natural actor-critic. In *Proceedings of the 16th European Conference on Machine Learning* (2005), pp. 280–291.

[82] Pickett, Marc, and Barto, Andrew G. Policyblocks: An algorithm for creating useful macro-actions in reinforcement learning. In *ICML* (2002), pp. 506–513.

[83] Poritz, A. Linear predictive hidden markov models and the speech signal. In *Proceedings of the Seventh IEEE International Conference on Acoustics, Speech, and Signal Processing* (1982), IEEE, pp. 1291–1294.

[84] Quattoni, Ariadna, Collins, Michael, and Darrell, Trevor. Conditional random fields for object recognition. In *In NIPS* (2004), MIT Press, pp. 1097–1104.

[85] Ramachandran, D., and Amir, E. Bayesian inverse reinforcement learning. *Proceedings of the 20th International Joint Conference on Artificial Intelligence* (2007).

[86] Rasmussen, Carl Edward. The infinite Gaussian mixture model. In *Advances in Neural Information Processing Systems 12* (2000), MIT Press, pp. 554–560.

[87] Riano, Lorenzo, and McGinnity, T.M. Automatically composing and parameterizing skills by evolving finite state automata. *Robotics and Autonomous Systems 60*, 4 (2012), 639–650.

[88] Rozo, L., Calinon, S., Caldwell, D. G., Jimenez, P., and Torras, C. Learning collaborative impedance-based robot behaviors. In *AAAI Conference on Artificial Intelligence* (Bellevue, Washington, USA, 2013).

[89] Schaal, S. Dynamic movement primitives–a framework for motor control in humans and humanoid robotics. *The International Symposium on Adaptive Motion of Animals and Machines* (2003).

[90] Schaal, S., Peters, J., Nakanishi, J., and Ijspeert, A. Learning movement primitives. *International Symposium on Robotics Research* (2004), 561–572.

[91] Şimşek, Özgür, and Barto, Andrew G. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *Proc. of the Twenty-First International Conference on Machine Learning* (2004), pp. 751–758.

[92] Şimşek, Özgür, and Barto, Andrew G. Skill characterization based on betweenness. In *Advances in Neural Information Processing Systems 22* (2008), pp. 1497–1504.

[93] Smart, W. D., and Kaelbling, L. P. Effective reinforcement learning for mobile robots. In *2002 IEEE International Conference on Robotics and Automation* (2002), pp. 3404–3410.

[94] Snel, Matthijs, and Whiteson, Shimon. Multi-task evolutionary shaping without pre-specified representations. In *GECCO* (2010), pp. 1031–1038.

[95] Sutton, Richard, Precup, Doina, and Singh, Satinder. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence 112* (1999), 181–211.

[96] Sutton, Richard S., and Barto, Andrew G. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[97] Tang, J., Singh, A., Goehausen, N., and Abbeel, P. Parameterized maneuver learning for autonomous helicopter flight. In *Proceedings of the IEEE International Conference on Robotics and Automation* (2010), IEEE, pp. 1142–1148.

[98] Teh, Yee Whye, Jordan, Michael I, Beal, Matthew J, and Blei, David M. Hierarchical dirichlet processes. *Journal of the American Statistical Association 101*, 476 (2006).

[99] Teh, Y.W., Jordan, M.I., Beal, M.J., and Blei, D.M. Hierarchical dirichlet processes. *Journal of the American Statistical Association 101*, 476 (2006), 1566–1581.

[100] Tellex, Stefanie, Kollar, Thomas, Dickerson, Steven, Walter, Mathew R., Banerjee, Ashis Gopal, Teller, Seth, and Roy, Nicholas. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence* (2011).

[101] Tesauro, G. Temporal difference learning and td-gammon. *Communications of the ACM 38*, 3 (1995), 58–68.

[102] Thomaz, Andrea L., and Breazeal, Cynthia. Reinforcement learning with human teachers: evidence of feedback and guidance with implications for learning performance. In *Proceedings of the 21st national conference on Artificial intelligence* (2006), pp. 1000–1005.

[103] Thrun, Sebastian, and Schwartz, Anton. Finding structure in reinforcement learning. In *Advances in Neural Information Processing Systems 7* (1995), MIT Press, pp. 385–392.

[104] Toussaint, Marc, Plath, Nils, Lang, Tobias, and Jetchev, Nikolay. Integrated motor control, planning, grasping and high-level reasoning in a blocks world using probabilistic inference. *IEEE International Conference on Robotics and Automation* (2010), 385–391.

[105] Vigorito, Christopher M., and Barto, Andrew G. Intrinsically motivated hierarchical skill learning in structured environments. *IEEE Transactions on Autonomous Mental Development 2*, 2 (2010).

[106] Wilcox, Ronald, Nikolaidis, Stefanos, and Shah, Julie. Optimization of temporal dynamics for adaptive human-robot interaction in assembly manufacturing. In *Proceedings of Robotics: Science and Systems* (2012).

[107] Willow Garage. Personal Robot 2 (PR2). http://www.willowgarage.com.

[108] Wilson, Aaron, Fern, Alan, Ray, Soumya, and Tadepalli, Prasad. Multi-task reinforcement learning: A hierarchical bayesian approach. In *In: ICML '07: Proceedings of the 24th international conference on Machine learning* (2007), ACM Press, p. 1015.

[109] Ziebart, B. D., Maas, A., Bagnell, J. D., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence* (2008).