Generative Models for Learning Robot Manipulation Skills from Humans

PhD thesis presented to
Computer and Communication Sciences Department
École Polytechnique Fédérale de Lausanne EPFL

Submitted in partial fulfilment requirements of PhD programme
by

Ajay Kumar Tanwani

accepted by the jury members:

Prof Pascal Frossard, jury president
Prof Hervé Bourlard, thesis director
Dr. Sylvain Calinon, thesis co-director
Prof Ken Goldberg, examiner
Prof Sethu Vijayakumar, examiner
Dr. Ronan Boulic, examiner

02 February, EPFL, 2018

Abstract

A long standing goal in artificial intelligence is to make robots seamlessly interact with humans in performing everyday manipulation skills. Learning from demonstrations or imitation learning provides a promising route to bridge this gap. In contrast to direct trajectory learning from demonstrations, many problems arise in interactive robotic applications that require higher contextual level understanding of the environment. This requires learning invariant mappings in the demonstrations that can generalize across different environmental situations such as size, position, orientation of objects, viewpoint of the observer, etc.

In this thesis, we address this challenge by encapsulating invariant patterns in the demonstrations using probabilistic learning models for acquiring dexterous manipulation skills. We learn the joint probability density function of the demonstrations with a *hidden semi-Markov model*, and smoothly follow the generated sequence of states with a *linear quadratic tracking* controller. The model exploits the invariant segments (also termed as sub-goals, options or actions) in the demonstrations and adapts the movement in accordance with the external environmental situations such as size, position and orientation of the objects in the environment using a task-parameterized formulation. We incorporate high-dimensional sensory data for skill acquisition by parsimoniously representing the demonstrations using statistical subspace clustering methods and exploit the coordination patterns in latent space. To adapt the models on the fly and/or teach new manipulation skills online with the streaming data, we formulate a non-parametric scalable online sequence clustering algorithm with Bayesian non-parametric mixture models to avoid the model selection problem while ensuring tractability under small variance asymptotics.

We exploit the developed generative models to perform manipulation skills with remotely operated vehicles over satellite communication in the presence of communication delays and limited bandwidth. A set of task-parameterized generative models are learned from the demonstrations of different manipulation skills provided by the teleoperator. The model captures the intention of teleoperator on one hand and provides assistance in performing remote manipulation tasks on the other hand under varying environmental situations. The assistance is formulated under time-independent shared control, where the model contin-

uously corrects the remote arm movement based on the current state of the teleoperator; and/or time-dependent autonomous control, where the model synthesizes the movement of the remote arm for autonomous skill execution. Using the proposed methodology with the two-armed Baxter robot as a mock-up for semi-autonomous teleoperation, we are able to learn manipulation skills such as opening a valve, pick-and-place an object by obstacle avoidance, hot-stabbing (a specialized underwater task akin to peg-in-a-hole task), screw-driver target snapping, and tracking a carabiner in as few as $4 - 8$ demonstrations. Our study shows that the proposed manipulation assistance formulations improve the performance of the teleoperator by reducing the task errors and the execution time, while catering for the environmental differences in performing remote manipulation tasks with limited bandwidth and communication delays.

keywords: generative models, learning from humans, hidden semi-Markov models, task-parameterized models, subspace clustering, Bayesian non-parametric mixture models, online learning, telerobotics, teleoperation

Résumé

Un objectif de longue date en intelligence artificielle est de permettre aux robots d'interagir sans difficulté avec les humains en accomplissant des gestes de manipulation de la vie de tous les jours. La programmation par démonstration ou l'apprentissage par imitation est une approche prometteuse pour franchir ce cap. De manière opposée à l'apprentissage direct des trajectoires des démonstrations, beaucoup de problèmes pour des applications robotiques interactives requièrent un plus haut niveau de compréhension contextuel de l'environnement. Cela nécessite d'apprendre les éléments invariants des démonstrations qui peuvent être généralisés à différentes situations relatives à l'environnement, comme la taille et l'orientation des objects, le point de vue de l'observateur, etc.

Dans cette thèse, nous extrayons les motifs invariants des démonstrations en utilisant des modèles d'apprentissage probabilistes pour acquérir des compétences de manipulation précise. Nous apprenons la densité de probabilité jointe des démonstrations avec un *hidden semi-Markov model* et suivons la séquence d'états générée avec un contrôleur de *suivi linéaire quadratique*. Le modèle exploite les motifs invariants des démonstrations et adapte le mouvement en fonction des situations de l'environnement externe, comme la taille, la position et l'orientation des objets dans l'environnement. Nous incorporons des données sensorielles de grande dimension pour l'acquisition de compétences, en représentant les démonstrations parcimonieusement, en utilisant des méthodes statistiques de partitionnement en sous-espaces, et en exploitant les motifs de coordinations dans ces sous espaces. Pour adapter les modèles à la volée et/ou pour apprendre des nouvelles compétences de manipulation en continu à l'aide des données, nous formulons un algorithme en ligne de partitionnement des séquences, non-paramétrique et adaptable. Ceci est réalisé à l'aide de modèles de mixtures bayésiennes non-paramétriques afin d'éviter les problèmes de sélection du modèle tout en assurant la traçabilité en présence de petites variations.

Nous exploitons les modèles génératifs développés pour des compétences de manipulation avec des véhicules opérés à distance via des communications satellites, caractérisées par la présence de délais dans la communication et possédant une bande-passante limitée. Un set de modèles génératifs paramétrés par la tâche est appris à partir des démonstrations des différentes compétences de manipulation fournies par le téléopérateur. Le modèle capture

d'une part l'intention du téléopérateur et, d'autre part, l'assiste pour effectuer des tâches de manipulations opérées à distance dans des situations où l'environnement est variable. L'assistance est formulée autour d'un contrôle partagé indépendant du temps, où le modèle corrige continuellement le mouvement du bras opéré à distance sur la base de l'état courant du téléopérateur; et/ou d'un contrôle autonome dépendant du temps, où le modèle synthétise le mouvement du bras opéré à distance pour l'exécution autonome de la tâche. Le robot Baxter, muni de 2 bras, est utilisé commen plateforme de test pour la téléopération semi-autonome. Avec ce robot, nous pouvons apprendre des compétences de manipulation comme ouvrir une valve, prendre et placer un object en évitant les obstacles, connecter des câbles avec des tuyaux, placer un tournevis dans un emplacement spécifique, ou accrocher un mousqueton en 4 à 8 démonstrations. Les techniques développées d'assistance à la manipulation améliorent les performances du téléopérateur et s'adaptent aux différences dans l'environnement pour exécuter des tâches de manipulation à distance avec une bande passante limitée et des délais dans la communication.

Mots clefs: modèles génératifs, apprentissage par interaction homme-machine, chaînes semi-Markoviennes, partitionnement en sous-espaces, modèles de mixtures non-paramétriques bayésiens, apprentissage en continu, télérobotique, téléopération.

Contents

Abstract (English/Français) **iii**

List of figures **xiii**

List of tables **xvii**

1 Robot Learning **1**

1.1 Robot Learning Problems: An Overview 3

1.1.1 Supervised Learning 4

1.1.2 (Inverse) Reinforcement Learning 5

1.1.3 Unsupervised Learning 6

1.2 Robot Learning for Teleoperation 10

1.2.1 DexROV: Dexterous Remotely Operated Vehicle Operations 12

1.2.2 Proposed Semi-Autonomous Teleoperation Approach 14

1.3 Thesis Contributions and Organization 15

2 Rewards-Driven Learning from Demonstrations **19**

2.1 Background and Related Work 20

2.2 Learning Reward Function(s) in Discrete Domains 24

2.2.1 Multiple Reward Functions 27
2.2.2 Transfer Learning in Optimal Policy Search 27
2.2.3 Grid World and Mini-Golf Examples 30
2.3 Reinforcement Learning in Continuous Domains 33
2.3.1 Actor-Critics with Experience Replay 35
2.3.2 Octopus Arm and Half-Cheetah Examples 38
2.3.3 Learning with Initial Policy 42
2.3.4 Decoding EEG Signals for Arm Control Example 42
2.4 Inverse Reinforcement Learning in Continuous Domains 49
2.4.1 Model-Based Learning for Trajectory Reward Function 49
2.4.2 Letter Writing Example 53
2.5 Conclusions 54

3 Task-Parameterized Generative Models 57

3.1 Encoding with Generative Models 58
3.1.1 Gaussian Mixture Model (GMM) 58
3.1.2 Hidden Markov Model (HMM) 60
3.1.3 Hidden Semi-Markov Model (HSMM) 62
3.1.4 Kalman Filter and Dynamic Bayesian Networks 64
3.2 Task-Parameterized Generative Models 65
3.2.1 Learning Model Parameters 66
3.2.2 Adapting Model Parameters in New Situations 68
3.2.3 Sampling from HSMM 70
3.3 Decoding with Linear Quadratic Regulator/Tracking (LQR/LQT) 72

3.3.1 Continuous LQR/LQT 73

3.3.2 Discrete LQR/LQT 75

3.4 Valve Opening Example 76

3.5 Conclusion 78

4 Scalable Generative Models in Latent Space 79

4.1 Subspace Clustering 80

4.1.1 High-Dimensional Data Clustering 81

4.1.2 Mixture of Factor Analyzers (MFA) Decomposition 82

4.2 Semi-Tied Mixture Model 84

4.2.1 Maximum Likelihood Parameter Estimation 86

4.2.2 Analysis of Semi-Tied Mixture Models 88

4.2.3 Whole Body Motion Capture Data - Chicken Dance Example 88

4.3 Task-Parameterized HSMM in Latent Space 89

4.3.1 Valve Opening Comparison 92

4.3.2 Pick-and-Place with Obstacle Avoidance Example 92

4.4 Conclusion 94

5 Bayesian Non-Parametric Online Generative Models 97

5.1 Background and Related Work 99

5.2 Problem Formulation under Small Variance Asymptotics (SVA) 101

5.3 SVA of DP-GMM 103

5.3.1 Dirichlet Process GMM (DP-GMM) 103

5.3.2 Online Inference in DP-GMM 104

5.4 Online DP-MPPCA 105

5.4.1 Dirichlet Process MPPCA (DP-MPPCA) 105

5.4.2 Online Inference in DP-MPPCA 107

5.5 Online HDP-HSMM 111

5.5.1 Hierarchical Dirichlet Process HSMM (HDP-HSMM) 111

5.5.2 Online Inference in HDP-HSMM 113

5.6 Scalable Online Sequence Clustering (SOSC) 115

5.6.1 Task-Parameterized Formulation of SOSC 117

5.7 Experiments, Results and Discussions 119

5.7.1 Synthetic Data 120

5.7.2 Tracking Screwdriver Target and Hooking Carabiner Examples . . . 122

5.8 Conclusions 128

6 Manipulation Assistance in Teleoperation 129

6.1 Teleoperation Scenario - An Illustrative Example 130

6.2 High Level Architecture 132

6.2.1 Cognitive Engine 132

6.2.2 Proxy Cognitive Engine 134

6.2.3 Communication Interfaces 135

6.3 Intention Recognition and Manipulation Assistance 135

6.3.1 Time-Independent Shared Control 138

6.3.2 Time-Dependent Autonomous Control 139

6.4 Comparison with Virtual Fixtures 140

6.5 Experiments, Results and Discussions 141

6.5.1 Task Performance Error 143

6.5.2 Robustness to Different Environments 144

6.5.3 Execution Time 145

6.6 Conclusion 145

7 Concluding Remarks and Future Directions **147**

Bibliography **155**

Curriculum Vitae **177**

List of Figures

1.1 Robot learning from human methodologies 3

1.2 Variants of the hidden semi-Markov model 8

1.3 Invariant task representation of the hidden semi-Markov model 9

1.4 DexROV high-level architecture 12

1.5 Shared and autonomous control modes for teleoperation 14

1.6 Manipulation skills learned with the Baxter robot 15

2.1 Learning multiple reward functions via inverse reinforcement learning 26

2.2 Value surface in optimal policy transfer 29

2.3 Grid world results for learning multiple reward functions 32

2.4 Mini-golf setup for learning different expert strategies with Barrett WAM . 33

2.5 Mini-golf results for the demonstrated and the learned strategies 34

2.6 Octopus arm action set 39

2.7 Octopus arm results of duration steps and goal reaching movement 40

2.8 Half-cheetah results of average rewards and running movement 41

2.9 Decoding goal from slow cortical EEG signals to drive KUKA robot arm . . 42

2.10 Evolving EEG channels activity in the time interval $[-1\ 1]$ seconds. 46

2.11 Decoding goal direction from EEG signals of a healthy subject 46

2.12 Policy evaluation of initial and optimized dynamical system 47

2.13 KUKA robot performing center-out reaching task in simulation 48

2.14 Letter writing setup with the KUKA robot. 52

2.15 Optimal policy generalization for writing letter 'a' under noisy conditions . 54

3.1 Gaussian mixture model encoding example 58

3.2 Graphical representation of a GMM, HMM and HSMM 64

3.3 Task-parameterized representation of hidden semi-Markov model 69

3.4 Sampling from HSMM and tracking with LQT 72

3.5 Open/close the valve with the Baxter robot 76

3.6 Baxter valve opening task reproduction results and HSMM encoding 76

3.7 Variance of the Baxter valve opening task model in coordinate systems . . . 77

4.1 Diagonal, full and MFA decomposition of covariance matrix representation . 82

4.2 Semi-tied mixture model encoding example 85

4.3 Semi-tied model parameters evaluation on chicken dance movement 89

4.4 Pairwise correlation comparison with standard and semi-tied GMM 90

4.5 Valve open/close model generalization with the Baxter robot 91

4.6 Pick-and-place by obstacle avoidance with the Baxter robot 93

4.7 Task-parameterized semi-tied HSMM evaluation on pick-and-place task . . . 93

5.1 Scalable online sequence clustering illustration 98

5.2 Parameter representation with non-parametric HSMM and non-parametric MPPCA 102

5.3 Clustering results with online DP-GMM and online DP-MPPCA 110

5.4 Ordering of data with SOSC 117

5.5 Scalable online sequence clustering with non-stationary data 119

5.6 Evolution of K and d_k with number of datapoints. 120

5.7 HSMM encoding results on non-stationary data 121

5.8 Scalable online sequence clustering with high-dimensional data 121

5.9 Online semi-autonomous teleoperation with the Baxter robot 123

5.10 Joint distribution of task-parameterized online learning example 123

5.11 Shared and autonomous control results with online teleoperation 124

5.12 HSMM graphical representation with online learning 124

6.1 Three step semi-autonomous teleoperation framework 131

6.2 Teleoperation setup to control the ROV 132

6.3 Cognitive engine simulator interface 134

6.4 ROS based system representation of the cognitive engine. 135

6.5 Publishers and subscribers in cognitive system 136

6.6 Virtual guided fixture for teleoperation 141

6.7 Screw-driver target tracking results by teleoperation 142

6.8 Open/close valve results by teleoperation 143

7.1 Spectrum of related robot learning directions. 152

List of Tables

2.1 Performance comparison of initial and optimized dynamical system 47

4.1 Performance analysis of tying model parameters 92

5.1 Performance comparison of batch and online learning 126

6.1 Performance comparison of different teleoperation modes 144

6.2 Average execution time of different teleoperation modes 145

Chapter 1

Robot Learning

Contents

1.1 Robot Learning Problems: An Overview 3

1.1.1 Supervised Learning 4

1.1.2 (Inverse) Reinforcement Learning 5

1.1.3 Unsupervised Learning 6

1.2 Robot Learning for Teleoperation 10

1.2.1 DexROV: Dexterous Remotely Operated Vehicle Operations 12

1.2.2 Proposed Semi-Autonomous Teleoperation Approach 14

1.3 Thesis Contributions and Organization 15

The world around us is going to change markedly with the use of robots assisting humans in everyday tasks. Robots are envisioned to be part of our daily lives in the form of wearable devices, search and inspection robots, medical robots, unmanned aerial vehicles/drones, autonomous driving cars, warehouse management systems and many other household and industrial applications. Despite popular imagination, majority of the robots today are limited to factories and assembly plants where they perform predefined tasks in a controlled environment. Such robots can be dangerous to the co-workers because of their large size and are typically separated by safety cages.

The next generation of robots relies on human-robot collaboration to overcome the adaptation barriers in the real world. Sensing, actuation and interaction technologies are at the core to facilitate natural collaboration between humans and machines. Better and cheaper sensors allow the robot to perceive the environment and respond accordingly. The actuation mechanisms and robot bodies are becoming more lightweight and flexible, allowing

'soft' interaction with the environment in a reliable and safe manner. More and more on-board computation is being performed on the cloud to make robots compact and modular for deployment in real-world environments.

Among all these advancements, the problem of motor control in robotics presents a basic ubiquitous challenge because of the complex robot dynamics, high dimensional sensory data and unstructured environmental conditions. Despite efforts to build good dynamic models and find computationally feasible closed loop controllers, it is difficult to predetermine the desired behaviour for every situation in terms of what the robot should do and what it should not. **Robot learning** is a fundamental characteristic to achieve true autonomy in robots. It enables a robot to acquire new skills from training examples representing the skill rather than directly programming them. Example of such skills include locomotion, grasping, object tracking and so on. Most of the work in robot learning is based on self-exploratory autonomous learning a.k.a reinforcement learning [Sutton 1992, Peters 2008, Kober 2013, Kormushev 2013], model learning [Nguyen-Tuong 2011] or imitation learning for skill acquisition [Schaal 2003, Argall 2009, Billard 2016].

Learning from humans, also known as imitation learning or programming by demonstration, provides a promising approach to facilitate robot learning in the most 'natural' way. Instead of hard coding/programming the robot to perform a task, the robot learns a new skill by observing a human performing the task. The goal of the robot is not to merely record and replay the demonstrated behaviour, but to be able to generalize across new situations of the task. The main challenges involved in robot learning from demonstrations include [Nehaniv 2004], 1) **what-to-learn** – acquiring meaningful data to represent the important features of the task from demonstrations, and 2) **how-to-learn** – learning a control policy from the features to reproduce the demonstrated behaviour.

We broadly address what-to-learn and how-to-learn problem in the context of robot learning from human demonstrations in this thesis. We are interested in quickly learning manipulation tasks from human demonstrations by breaking them down into meaningful segments and sequencing them together to generalize across different environmental situations. The problem of segmenting the demonstrations has been studied in different contexts in various scientific communities, such as action detection or activity recognition in computer vision (see [Spriggs 2009, Yeung 2016] for example); options framework in the reinforcement learning community (see [Stolle 2002, Krishnan 2017] for example); sequencing primitives in robotics (see [Pastor 2012, Manschitz 2016, Medina R. 2017] for example). We are interested in the generative modeling aspect of the demonstrations for segmentation, recognition, and synthesis of robot manipulation tasks. We study these models in the context of task-parameterized models, under which the demonstrations are

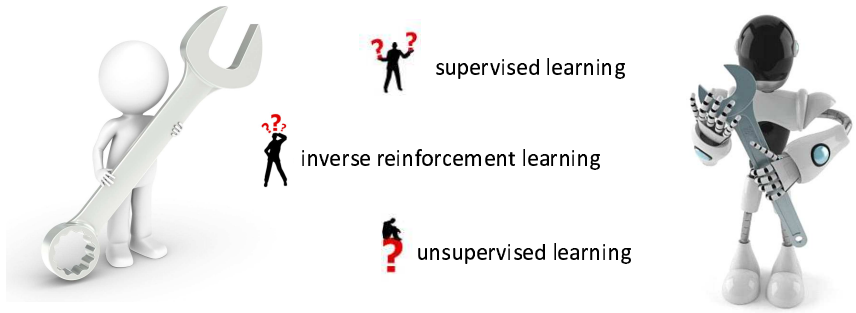


Figure 1.1: Robots can acquire manipulation skills from human demonstrations using supervised learning, inverse reinforcement learning and/or unsupervised learning. *(images adapted from https://shutterstock.com)*

observed in different coordinate systems describing virtual landmarks or objects of interest and the model is adapted according to the environmental changes in a systematic manner [Wilson 1999, Calinon 2016, Tanwani 2016a]. In this thesis, we develop a family of these models for acquiring dexterous manipulation skills. We are interested in recognizing the intention of the user in the demonstrations and subsequently, exploit these generative models to perform dexterous manipulation with robots that are far away from us by providing assistance to the user. Our application specific goal is to perform these dexterous manipulation activities remotely in challenging underwater environments, inspired by how space telerobotics have enabled us to operate the Curiosity rover on Mars from the control center on Earth via satellite communication.

The chapter is organized as follows: we first provide an overview of robot learning from demonstrations in the context of this thesis. We then explain the application scenario of our work for performing remote manipulation tasks by semi-autonomous teleoperation. Finally, we provide an outlook to the remaining chapters and summarize the main contributions of the thesis.

1.1 Robot Learning Problems: An Overview

Learning from demonstrations can be formulated in different ways depending upon the underlying assumptions of the environment and the model. Let us denote $\boldsymbol{\xi}_t \in \mathbb{R}^D$ as the state of the environment at time t. The state may represent the visual observation, kinesthetic data such as the pose and the velocities of the end-effector of the human arm, haptic information, or any arbitrary features defining the task variables of the environment. Given a set of datapoints $\{\boldsymbol{\xi}_t\}_{t=1}^T$ over T time steps representing N demonstrations

of performing the task under different initial conditions of the environment, the goal of learning from demonstrations is to find the policy or function that outputs the control input $\boldsymbol{u}_t \in \mathbb{R}^m$ at each time step such that the robot is able to execute the underlying task in a smooth manner. The control policy can be a function of the current state of the environment $\boldsymbol{\xi}_t$, and/or time t in an implicit or explicit manner. The problem of learning this control policy from demonstrations can be formalized in three main ways depending upon the underlying assumptions of the demonstrations (see Fig. 1.1).

1.1.1 Supervised Learning

In supervised learning problems, the learner is given training examples of the form of input-output pairs $\{(\boldsymbol{\xi}_t, \boldsymbol{y}_t)\}_{t=1}^T$ where the output $\boldsymbol{y}_t$ may be discrete for classification problems or continuous for regression problems. The input samples $\boldsymbol{\xi}_t$ are assumed to be drawn from a fixed probability distribution $\mathcal{P}(\boldsymbol{\xi}_t)$ and are mapped to predict the output samples $\boldsymbol{y}_t$ by the function $f : \boldsymbol{\xi}_t \rightarrow \boldsymbol{y}_t$. The mapping can be deterministic, $\boldsymbol{y}_t = f(\boldsymbol{\xi}_t)$, or the output samples can be stochastically obtained by computing $f(\boldsymbol{\xi}_t) + \varepsilon$ where ε is a white noise added to the output.

Without loss of generality, we represent the function learned from training examples by regression $\boldsymbol{y}_t = f(\boldsymbol{\xi}_t; \boldsymbol{\theta})$ with parameter vector $\boldsymbol{\theta} \in \mathbb{R}^K$. The parameters compactly represent the relationship between the input and the output samples that allow prediction for new unseen input data. Most common methods to estimate the function parameters from training examples including Locally Weighted Regression (LWR) [Atkeson 1997b], Locally Weighted Projected Regression (LWPR) [Vijayakumar 2000], Gaussian Mixture Regression (GMR) [Ghahramani 1994], Support Vector Regression [Smola 2004], Gaussian Process Regression [Rasmussen 2006]. An interested reader can see a unifying review of these regression approaches in [Stulp 2015]. Two common encoding representations most commonly used with supervised learning include

1) **Time-Indexed Reference Trajectory:** The input sequence can be described in terms of a time-indexed reference trajectory [Peters 2008]. The reference trajectory can be translated into the control input $\boldsymbol{u}_t$ using a proportional-derivative (PD) controller at each time step [Miyamoto 1996], $\boldsymbol{u}_t = \boldsymbol{K}(\hat{\boldsymbol{\xi}}_t - \boldsymbol{\xi}_t)$, where, $\boldsymbol{K} \in \mathbb{R}^{m \times D}$ represents the stiffness and damping gains and $\hat{\boldsymbol{\xi}}_t = f(t; \theta)$ is the learned desired reference trajectory.

2) **Dynamic Movement Primitives:** The demonstrations are formulated as a set of non-linear differential equations with well-defined attractor dynamics [Ijspeert 2002]. A DMP consists of a *transformation system* and a *canonical system*, where the transformation

system exploits the attractor properties of a dynamical system for discrete or rhythmic behaviour and the canonical system determines the 'phase' of the system functioning as a substitute of time. The two systems are coupled with a non-linear forcing term that modifies the attractor landscape of the dynamic system according to the desired trajectories. The advantage of this transformation is that time is implicitly embedded in the phase which can be manipulated easily to control the evolution of the system (for example, adding coupling terms, phase resetting etc.) [Ijspeert 2013].

1.1.2 (Inverse) Reinforcement Learning

Humans are able to analyze their decisions in an abstract way based on rewards and penalties. It is often easier to describe the demonstrations in terms of a score reflecting the performance criterion of a skill, than to directly encode the demonstrations [Mnih 2015]. For example, in order to learn the skill of swinging up a pendulum and balancing on a hand, it may be easier to assign a positive score only when the pendulum is balanced.

Such problems are formalized in the context of RL where the goal is to find a policy such that the sum of scores assigned to the states visited by the learner are maximized over some time horizon [Sutton 1992]. The key difference in this context compared to supervised learning is that there is no fixed distribution $\mathcal{P}(\boldsymbol{\xi}_t)$ from which the datapoints are drawn and the learner is required to choose which datapoints to visit. Moreover, the goal in reinforcement learning is to find the sequence of datapoints that maximize the sum of scores or rewards received while visiting the datapoints, compared to minimizing the loss function over the entire space in supervised learning. Note that solving the problem now requires knowledge about the transition dynamics of the environment as well in order to choose better control actions and visit datapoints with higher rewards.

When the human behaviour encapsulates such sequential decision making based on rewards, it makes more sense to transfer the underlying reward function to the robot, instead of directly imitating the demonstrated behaviour. The whole paradigm of RL is based on the assumption that reward function – not the policy – is the most succinct and transferable representation of a skill. Inverse reinforcement learning is motivated by the difficulty to specify the reward function in reinforcement learning [Ng 2000, Abbeel 2004, Ziebart 2008, Neu 2012]. Even when a reward function is difficult to describe exactly, it is usually intuitive to decide what the reward function must depend on as there are often multiple criteria the learner should optimize for in the policy, e.g, in a car driving example, avoid collision with other cars and pedestrians while driving as fast as possible and so on. Combining these multiple desired criteria into one scalar reward function is often non-trivial

and results in a waste of time and data samples required to hand-tune the parameters. Compared to supervised learning approach of penalizing deviations from demonstrations using some regression technique, IRL extracts the reward function to acquire a compact representation of the skill in the demonstrations and then produces the optimal policy to generalize in new situations.

Although attractive for better generalization in new situations for sequential decision making problems, extracting the true reward function from demonstrations is challenging due to the ill-posed nature of the problem. Many reward functions lead to the same optimal demonstrations, for example, all demonstrations are optimal for zero reward function. Moreover, it is well-known that humans vary widely in performing sequential decision-making tasks, possibly differing in their intentions or ways of gauging task-dependent features. In chapter 2, we address the problem of rewards-driven learning from multiple demonstrators with different underlying intention(s) or strategies of performing a task [Tanwani 2013b].

1.1.3 Unsupervised Learning

In the unsupervised learning case, there are no labels or targets for the training samples in the demonstration. The goal of unsupervised learning is to model the underlying probability distribution or density function of the demonstrations $\mathcal{P}(\boldsymbol{\xi}_t)$ from which the datapoints are sampled in order to reveal the structure in the demonstrations.

With the increasing amount of high-dimensional sensory data and multimodal interfaces for skill acquisition in robotics, unsupervised algorithms aim to discover the patterns in the data on their own that can be used for reasoning, decision making, prediction and so on. Note that unsupervised learning is a considerably harder problem than supervised learning since there are no target output to learn from. Unsupervised learning is most commonly used to group the demonstrations into different segments/clusters, dimensionality reduction and/or to learn the association rules from the demonstrations [Weber 2000].

Generative models are most commonly used for unsupervised learning where the goal is to train a model that can generate the data like the observed demonstrations. These models learn the joint probability distribution of the data, in contrast to discriminative models that directly learn the conditional probability distribution of the demonstrations as in supervised learning problems. The most commonly used generative models include Gaussian mixture models, Hidden Markov models [Rabiner 1989], Naive Bayes [Friedman 1997], latent space models (Principal Component Analysis, Factor Analysis) [Fodor 2002], Re-

stricted Boltzmann machine [Salakhutdinov 2007], and Generative Adversarial Networks [Goodfellow 2014]. All these models can be seen as an instance of a basic generative model [Roweis 1999]. In contrast, common discriminative models include Logistic regression, Support Vector Machines, Maximum Entropy Models, Conditional Random Fields, and Neural Networks. Other approaches such as semi-supervised learning are also gaining popularity now in problems where only a few datapoints are associated with target outputs [Zhu 2009].

The major focus of this thesis is on learning and reproduction of manipulation skills with generative mixture models. **Gaussian mixture models (GMMs)** are widely used to encode local trends in the demonstrations. For example, the demonstrations can be encoded as a state-dependent autonomous dynamical system (independent of passing time) of the form $\dot{\boldsymbol{\xi}} = f(\boldsymbol{\xi};\boldsymbol{\theta})$ [Khansari-Zadeh 2011]. **Hidden Markov models (HMMs)** encapsulate spatial and temporal information by augmenting a GMM with latent variables that sequentially evolve over time in the demonstrations. HMMs are widely used for time series/sequence analysis in speech recognition, machine translation, DNA sequencing, robotics and many other fields [Rabiner 1989]. HMMs have been typically used for recognition and generation of the movement skills in robotics [Asfour 2008, Calinon 2010, Lee 2010b, Vakanski 2012]. Several shortcomings of encoding with HMMs have been pointed out in the literature, including: 1) how to bias learning towards models with longer self-dwelling states, 2) how to robustly estimate the parameters with high-dimensional noisy data, 3) how to adapt the model with newly observed data, 4) how many states should the model possess.

In this thesis, we present several alternatives to make robot learning with generative mixture models suitable for encoding and decoding of real-world robot manipulation tasks under varying environmental situations.

We build upon **Hidden semi-Markov models (HSMMs)** that replace the self-transition probabilities of staying in a state with an explicit model of state duration [Yu 2010]. This helps to adequately bias the generated motion with longer state dwell times for skill acquisition. We show representations with different sensory encodings and exploit the variability in the demonstrations with a **linear quadratic tracking** (LQT) controller during reproduction [Tanwani 2016a].

We perform segmentation and dimensionality reduction simultaneously with **subspace clustering** methods to impose a parsimonious structure on the covariance matrix and reduce the number of parameters that can be robustly estimated. We learn the model in latent space based on statistical decomposition of the covariance matrix, and exploit coordination patterns and synergistic directions for scalable and efficient skill encoding of

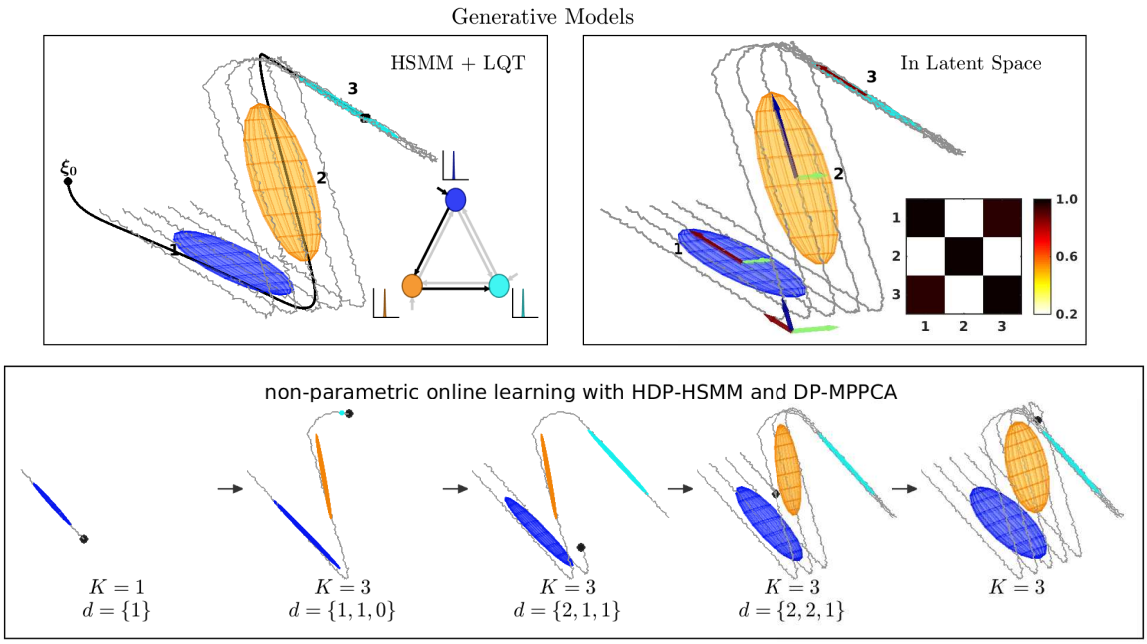


Figure 1.2: Generative model formulations developed in the thesis for robot learning problems on Z-shaped data: *(top-left)* demonstrations encoded with a hidden semi-Markov model and decoded with a linear quadratic tracking controller (Chap. 3), *(top-right)* the movement data is encoded in latent space for parsimonious representation and better generalization (Chap. 4), *(bottom)* Bayesian non-parametric scalable online sequence clustering of the demonstrations (Chap. 5).

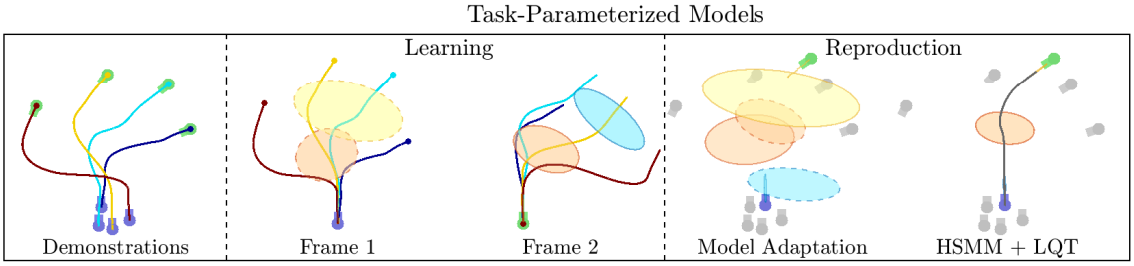


Figure 1.3: Task-parameterized formulation of generative models for robot learning (Chap. 3 – 5). The demonstration on left are observed from the coordinate systems that move with the object (starting in purple position and ending in green position in each demonstration) and the generative model is learned in the respective coordinate systems. The model parameters in respective coordinate systems are adapted to the new unseen object positions by computing the products of linearly transformed Gaussian mixture components. The resulting HSMM is combined with LQT for smooth retrieval of manipulation tasks.

the demonstrations [Tanwani 2016c].

Bayesian non-parametric treatment of these clustering problems provides flexibility in model selection by maintaining an appropriate probability distribution over parameter values with infinite number of states. Although attractive for encapsulating *a priori* information about the task, the computational overhead of existing sampling-based and variational techniques for inference limit the widespread use of these models. Recent analysis of Bayesian non-parametric mixture models under *small variance asymptotic* (SVA) limit has led to simple deterministic models that scale well with large size applications [Kulis 2012, Broderick 2013]. For example, as the variances of the mixture model tend to zero in a GMM, the probabilistic model converges to its deterministic counterpart, k-means, or to its non-parametric Dirichlet process (DP) version, DP-Means [Kulis 2012]. We formulate online learning algorithms of Bayesian non-parametric mixture models, namely Dirichlet process Gaussian mixture model (DP-GMM), Dirichlet process mixture of probabilistic principal component analysis (DP-MPPCA), and hierarchical Dirichlet process hidden semi-Markov model (HDP-HSMM). Applying SVA limit yields a *scalable online sequence clustering* (SOSC) algorithm which allows the model to readily adapt online with streaming high-dimensional data [Tanwani 2016b, Tanwani 2016c].

An overview of these models can be seen in Fig. 1.2. A task-parameterized formulation of these models is used to make the model invariant with respect to changing environmental situations such as position/size/orientation of the objects (see Fig. 1.3 for an overview of the invariant task representation method). This allows us to efficiently encode and synthesize motion for skill acquisition with a few expert demonstrations in an invariant manner.

1.2 Robot Learning for Teleoperation

Teleoperation provides a low cost solution to offload tedious work from humans and reach distant and/or hazardous environments. Teleoperated robots are going to increasingly assist humans in performing everyday life tasks as diverse as minimally invasive surgeries, security/surveillance, telepresence, warehouse management, remote patient monitoring, inspection/exploration in deep underwater or space missions.

Teleoperated robots are traditionally based on master-slave architecture where the teleoperator (master) transmits position/force to the robot (slave) in a unilateral mode, or transmits and receives position/force via bilateral communication (see [Niemeyer 2008] for a detailed review). Bilateral teleoperation uses a haptic interface to make the operator feel a particular impedance relative to the slave position or the force recorded between the slave and the environment. Despite the simple mechanism, teleoperation requires skilled personnel to remotely operate the robot, while having limited access to the controllable degrees of freedom and the sensory feedback. Moreover, stability issues arise in handling environmental uncertainty with communication delays between the teleoperator and the robot. This has motivated several control theoretic solutions such as scattering approach, wave variables, passivity based control, multichannel feedback and model prediction based control to deal with delayed force reflections [Hokayem 2006]. Modern day teleoperation systems use additional interfaces such as exoskeleton and/or head mounted display to increase the sense of *telepresence* in performing the task [Sheridan 1995, Zhang 2017].

The teleoperator controls the remote robot using either: 1) direct control, 2) shared control, or 3) supervisory control. Direct teleoperation lacks the autonomy/intelligence to assist the operator and the remote robot simply mimics the movement of the teleoperator. Shared control fine-tunes/complements the continuously streamed teleoperator data by local sensory feedback on the remote side. For constrained manipulation tasks, *virtual fixtures* have been used to reduce the operator workload by influencing the robot motion along desired paths [Rosenberg 1993, Abbott 2007]. Supervisory or autonomous control gives local autonomy to the remote robot to execute manipulation tasks in the presence of large communication delays. It makes use of predictive displays and high-level symbolic commands of atomic structure (such as reach, grasp, etc.) to breakdown a task in smaller subtasks [Sheridan 1992, Yoerger 1987].

Robot learning from demonstrations is a promising approach to assist humans in performing daily life tasks. In this context, advancing autonomy in teleoperation addresses two main problems: 1) predicting the operator's intent while performing the task, and 2) deciding how to assist the teleoperator. Both aspects are closely related in cooperative robots

for human-robot collaboration [Rozo 2016], and in general describe what-to-imitate and how-to-imitate problems in programming by demonstration. Depending upon how the word *intention* is phrased, a vast amount of literature exists to encapsulate the behaviour of the operator and subsequently, decode it for assistance. For example, predicting the user intent can be posed as a classification problem of reaching a particular goal position in a predefined set of goals [Yu 2005]. Alternatively, the user may be assumed to maximize an unknown reward function, to be ascertained by inverse reinforcement learning (IRL). Dragan and Srinivasa formulated a policy blending mechanism to combine the teleoperator intention with the robot movement using IRL [Dragan 2013]. In cognitive science, Bayesian models are more commonly used to incorporate uncertainty in decoding the user behaviour. Hauser in [Hauser 2013] inferred the type of task performed by the user with a Bayesian Gaussian mixture auto-regression framework, and followed the predicted trajectory with a cooperative motion planner.

Generative models such as Hidden Markov Models (HMMs) have been widely used to interpret human intention as performing a discrete set of tasks/subtasks with common low level sensory observations. The use of hierarchical representations [Aarno 2008], or sets of dynamic models for the subtasks sequenced together with a Markov chain [Pentland 1999], have been investigated to describe several human behaviours. Li *et al.* used virtual fixtures with a HMM to segment if the user intends to follow a periodic motion curve, not follow the curve or stay idle [Li 2003]. Nolin *et al.* in [Nolin 2003] investigated settings of discrete compliance levels with a HMM, namely $\{toggle, fade, hold\}$, to assist the user in following the virtual fixture based on his/her demonstration. Roila *et al.* presented probabilistic virtually guided fixtures for assistance [Raiola 2015]. Medina *et al.* perform task segmentation with an HMM and incrementally update its parameters during reproduction to progressively increase the collaborative role of the robot in performing the task [Medina 2011]. Wang *et al.* infer the probability distribution over intentions from the human observations in the latent state of a Gaussian process dynamical model [Wang 2013]. Maeda *et al.* recognize the phase/stage of human movement from intermittent observations under different possible speeds and plan a collaborative trajectory for the robot [Maeda 2015].

Improving autonomy in teleoperation, however, poses all kind of challenges to the existing techniques due to limited bandwidth, communication latency, and environmental differences between the teleoperator and the remote sites. Advancing the state-of-the-art in teleoperation is the central focus of many research programs, including DARPA Robotics challenge and NASA Space Robotics Challenge, and is the main application scenario of this thesis.

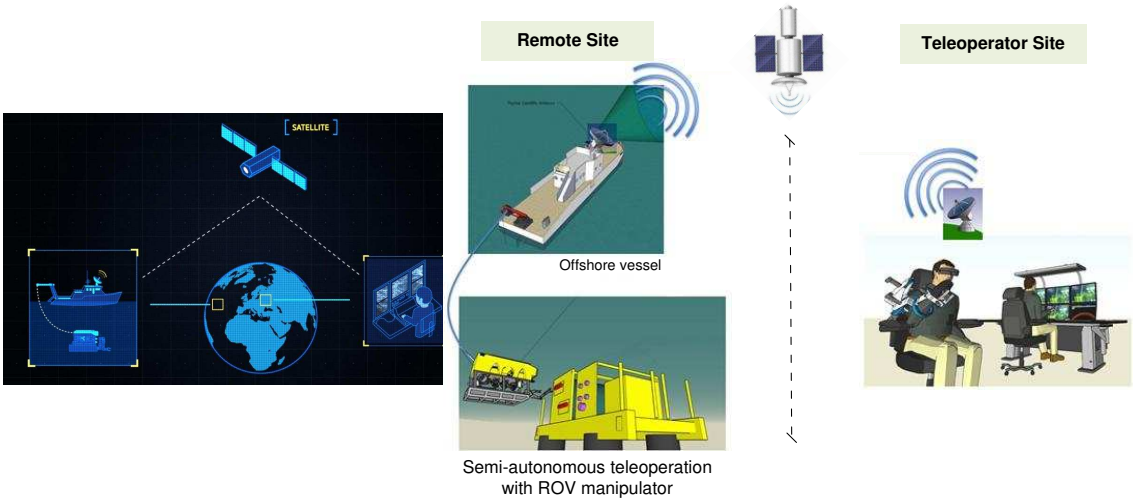


Figure 1.4: DexROV high-level architecture *(left)* the teleoperator and the remote sites are located in different parts of the world and linked over satellite communication; *(right)* the teleoperator performs the skill in the control center with a wearable exoskeleton in the virtual environment and the remote ROV manipulator arm performs the skill in a semi-autonomous manner to mitigate the effect of communication delays.

1.2.1 DexROV: Dexterous Remotely Operated Vehicle Operations

Many useful robotics applications require performing tasks in environments that are not friendly for humans. One typical example is underwater activities, ranging from inspection and maintenance of underwater cables and pipelines, to underwater archaeology and marine biology. There has been a boom in underwater remotely operated vehicles (ROVs) over the past few years. Nonetheless the cost of using ROVs is still prohibitively high for wider adoption, as currently ROV usage still requires substantial off-shore support to handle and operate the robotic platform. One of the main limiting factors is that a large off-shore crew is required to supervise and teleoperate the ROV directly from the support vessel. This is mainly due to the need of direct teleoperation, i.e. the operator receives visual feedback from an array of cameras on the ROV and accordingly uses a set of buttons, knobs and joysticks to guide the motion of all, body and arm(s), degrees-of-freedom (DoF) of the ROV. This cost can be reduced by moving the support and teleoperation team to an on-shore facility and communicating with the ROV remotely. Current satellite communications technology suffers from large latencies and deems traditional direct ROV teleoperation infeasible.

The European Commission H2020 project DexROV – Dexterous ROV operations in the presence of Communication Latencies – aims to work out more cost-efficient and time-efficient ROV operations by: 1) far distance teleoperation with fewer off-shore personnel in-

volving variable communication latencies to mitigate, and 2) dexterous manipulation assistance capabilities benefiting from context specific human skills [Gancet 2015, Gancet 2016]. Existing ROV based operations are preferred to diver based operations as the depth at which divers can work rarely exceeds 100 meters, however, the operations performed by ROVs are rather limited. We seek to teleoperate ROVs at a depth of around 1300 meters while using human expertise to perform dexterous operations. The overall conceptual architecture is shown in Fig. 1.4. The setup is distributed on two sides, described in the next two subsections.

1.2.1.1 Remote Site

On the offshore side, a vessel (with reduced crew) is tied to a ROV by a tether and equipped with a satellite communication link. The ROV is provided with an underwater perception system that facilitates: 1) online accurate and reliable navigation of ROV and mapping of the environment to facilitate safe operations relative to subsea installations and other structures [Pfingsthorn 2016], and 2) online object detection, recognition, modelling, and tracking for semi-autonomous teleoperation [Pathak 2010]. The ROV is mounted with two 6 degrees of freedom electric manipulator arms each possessing a hand with three finders and 2 active degrees of freedom for grasping a wide range of objects. Moreover, the arms are equipped with a set-based control to incorporate multiple task priorities while performing the task [Antonelli 2015, Simetti 2017]. Before performing a manipulation task, the ROV stations in front of the manipulation test-bed to enable the teloperator to remotely perform the task in a stationary manner.

1.2.1.2 Teleoperator Site

On the onshore teleoperator side, the monitoring and control center allows remote supervision and teleoperation of ROV to perform dexterous manipulation tasks. The teleoperator is mounted with intuitive sensory interfaces including a force-feedback exoskeleton and virtual reality headset to immerse the teleoperator in the actual environment. The communication between onshore and offshore side is handled using satellite communication with Cobham Sailor 800 3-axis stabilized and tracking Ku band antenna. The upper limit of forward (onshore to offshore) bandwidth is 256 Kbps and return (offshore to onshore) bandwidth is 768 Kbps hosted by VSAT communication provider Omniaccess [Gancet 2016]. Large communication delays with satellite communication render direct teleoperation infeasible and need semi-autonomous capabilities of the remotely operated vehicle to carry out manipulation tasks.

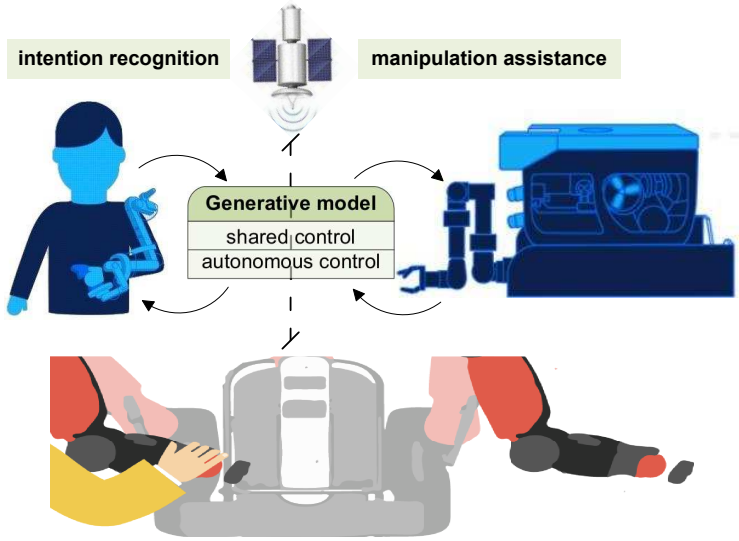
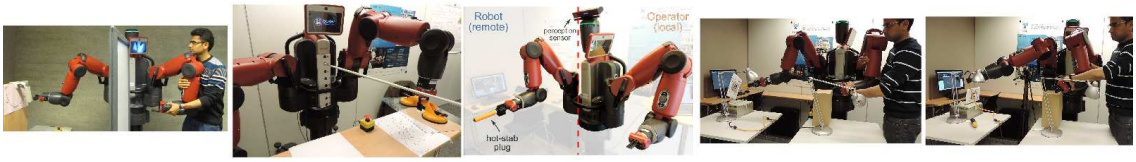


Figure 1.5: *(top)* generative model locally recognizes the intent of the teleoperator and provides manipulation assistance, *(bottom)* teleoperation mock-up with the two-armed Baxter robot where one arm is used as input device for the teleoperator and other arm is used to perform remote manipulation tasks.

1.2.2 Proposed Semi-Autonomous Teleoperation Approach

With limited communication bandwidth and communication latency in transmitting and receiving data between the teleoperator and remote sites, our goal is to develop semi-autonomous capabilities to the remote arm so that it can continue to perform the assigned task on its own till further communication is established with the teleoperator. We develop a novel teleoperation solution from the demonstrations provided by the teloperator within which no direct teleoperation is required but control is locally handled (onboard) using a probabilistic representation of task/skill primitives. Such a representation can adapt to changing task parameters and is robust against intermittent communication. We apply our algorithms to create a library of skills models constituting the *cognitive engine* that is used to detect the intention of the teleoperator and subsequently, assist the teleoperator in performing remote manipulation tasks. The learned model parameters are first transmitted to both the teleoperator side and the robot side. The task parameters on the robot side can then update the model of the skill at fast pace by local sensing, without requiring the transmission of this change to the teleoperator.

The cognitive engine receives input from the teleoperator that drives the robot arm in the local virtual reality environment without having to worry about the transmission delays. On the remote site, the generative model is used to synthesize the arm movement without disruption and limited synchronization between the teleoperator and the remote site. The



valve opening | pick-and-place with obstacle avoidance | hot stabbing | screw-driver snapping | carabiner hooking

Figure 1.6: Examples of manipulation skills learned with the task-parameterized generative models.

model locally anticipates which actions and/or regulation feedback policies to adopt until a new command or sensory information is available.

We formulate time-independent shared control and time-dependent autonomous control formulations of the developed task-parameterized generative models to assist the teleoperator in performing remote manipulation tasks in Chap. 6. In the time-independent shared control mode, the control is shared between the teleoperator and the remote site and the model corrects the remote arm movement based on the current state of the teleoperator; whereas in the time-dependent autonomous control mode, we sample the sequence of states to be visited for the next time horizon and the model generates the movement of the remote arm for autonomous task execution. Note that we only provide assistance in regions that have been explored by the demonstrations during the learning phase.

We use the two-armed Baxter robot as a mock-up of the teleoperation system, i.e., one arm becomes the input device for the teleoperator, and the other one is used for performing the manipulation task. The operator controls/teleoperates the remote arm with a simulated delay using the other arm by getting visual feedback from the remote arm. A set of kinesthetic demonstrations of the teleoperator is used to teach the robot how to perform each task. We leverage upon probabilistic generative models to understand the intention of the teleoperator and assist the movement on the robot side under varying environmental situations in performing a set of skills. Our key performance indicators are to: 1) reduce the cognitive load of the teleoperator for routine tasks, 2) cater for environmental difference and communication delays in performing remote manipulation tasks, and 3) reduce the time of operation of the teleoperator in performing the task.

1.3 Thesis Contributions and Organization

Below, we summarize the main contributions and describe the work flow of this thesis:

Learning Multiple Skills with IRL: We first learn multiple skills from demonstrations in the inverse reinforcement learning setting in Chapter 2 where the demonstrations are driven by different reward functions. We enclose the demonstrations in a convex set of optimal deterministic policies and use transfer of knowledge to bootstrap the learning of new skill. The challenges in scaling the approach to high-dimensional continuous spaces lead us to encoding the skills with generative models as density estimation problem in the subsequent chapters. The work on making reinforcement and inverse reinforcement learning suitable in continuous domains is carried out with external collaborators for various robot learning applications and can be read independently from the remainder of the thesis.

Task-Parameteriezd Hidden Semi-Markov Model for Skill Acquisition: In Chapter 3, we present hidden semi-Markov models for learning and reproduction of robot manipulation tasks. Task-parameterized formulation of the model allows us to systematically adopt the model parameters with changing environmental situations such as position/orientation/size of the objects, and generalize better in previously unseen situations. The planned movement sequence from the model is combined with linear quadratic tracking for autonomous reproduction in changing environmental situations.

Scalable Generative Models in Latent Space: We investigate parsimonious representation of the demonstrations in latent space for robust learning and adaptation of robot manipulation tasks in Chapter 4. We make use of the spatial and temporal correlation in the data by tying or decomposing the covariance matrices of the mixture model with common synergistic directions/basis vectors, instead of estimating full covariance matrices for each cluster in the mixture. This allows us to exploit the coordination patterns along important synergistic directions and reuse the discovered synergies in different parts of the skill having similar coordination patterns. The resulting task-parameterized generative model is data efficient and offers better generalization with much less parameters than mixture models with full covariance matrices.

Bayesian Non-Parametric Online Learning with Generative Models: In order to learn new manipulation skills on the fly from the demonstrations and/or to adapt the above generative models online with the streaming data, we analyse the above Bayesian non-parametric formulations of the mixture models under small variance asymptotics in Chapter 5. The analysis yields a non-parametric task-parameterized scalable online sequence clustering algorithm for learning and reproduction of high-dimensional robot manipulation skills from streaming data.

Manipulation Assistance in Teleoperation: These task-parameterized generative models constitute the core of *cognitive engine* in DexROV, responsible for providing as-

sistance to the teleoperator for performing remote manipulation tasks. We present probabilistic formulations of the model to capture the intention of the teleoperator and subsequently, assist the teleoperator by time-independent shared control and/or time-dependent autonomous control formulations of the model in Chapter 6. In the shared control mode, the model corrects the remote arm movement based on the current state of the teleoperator; whereas in the autonomous control mode, the model generates the movement of the remote arm for autonomous skill execution. We evaluate the performance of our approach under several key performance indicators to quantify the improvement of the teleoperator in performing remote manipulation tasks.

Our developed approaches allow us to perform challenging manipulation tasks by semi-autonomous teleoperation in as few as $4 - 8$ demonstrations, including opening a valve, pick-and-place an object by avoiding obstacles, inserting hot-stab plug into a receptacle, tracking a moving target with a screwdriver, and hooking a carabiner (see Fig. 1.6). These manipulation tasks constitute the routine tasks performed manually by the teleoperator with underwater ROVs.

Chapter 2

Rewards-Driven Learning from Demonstrations

Contents

2.1 Background and Related Work **20**
2.2 Learning Reward Function(s) in Discrete Domains **24**
2.2.1 Multiple Reward Functions 27
2.2.2 Transfer Learning in Optimal Policy Search 27
2.2.3 Grid World and Mini-Golf Examples 30
2.3 Reinforcement Learning in Continuous Domains **33**
2.3.1 Actor-Critics with Experience Replay 35
2.3.2 Octopus Arm and Half-Cheetah Examples 38
2.3.3 Learning with Initial Policy 42
2.3.4 Decoding EEG Signals for Arm Control Example 42
2.4 Inverse Reinforcement Learning in Continuous Domains **49**
2.4.1 Model-Based Learning for Trajectory Reward Function 49
2.4.2 Letter Writing Example 53
2.5 Conclusions **54**

Reinforcement learning (RL) has found a large variety of applications to describe sequential decision making problems in which the reward/feedback from the environment is used to guide the action-selection process of the robot/learner. In contrast, inverse reinforcement learning (IRL) aims to find the reward function given the optimal behaviour of the learner in the environment. Given the optimal set of demonstrations and the underlying transition

dynamics model, the goal is to find the reward function that is being maximized in the demonstrations. Contrary to supervised learning problems, IRL problems leverage upon the problem structure in the reward function and transition model to estimate the control policy that drives the demonstrator.

IRL provides an interesting approach to learning task models from demonstrations by first inferring the intention of the demonstrator in the form of a reward function and then optimizing the reward function to derive the control policy for the robot. The paradigm of IRL is based on the assumption that the reward function – not the control policy (demonstrations) – is the most succinct and generalizable representation of the task.

In this chapter, we first briefly review the fundamentals and important methods of solving RL/IRL problems. The challenges in its design will lead us to present our methodology of learning from multiple demonstrations which can incrementally incorporate multiple reward functions and transfer knowledge to speed up the learning of multiple strategies for the robot to perform the task. Following that, we move to RL/IRL in continuous domains and present solutions to address existing limitations in rewards-driven learning from demonstrations paradigm.

2.1 Background and Related Work

Reinforcement learning is a trial-and-error method driven mainly by feedback from the environment or rewards [Sutton 1998]. The basic procedure of reinforcement learning starts with the environment being initially in some state. The learner, or the agent/robot, interacts with the environment by taking an action (the only way an agent can interact with the environment). The state of the environment changes due to that action and the agent gets some reward as feedback. Then the agent observes the new state and repeats the procedure again. During the successive iterations, the agent also tries to modify its policy of interaction with the world with an aim to maximize the rewards obtained. Since the agent learns to interact with the environment - this type of learning is in essence online.

The interaction between the environment and the learner is captured in a Markov Decision Process (MDP) represented by a tuple $< \boldsymbol{S}, \boldsymbol{A}, \mathcal{P}_{sa}, \gamma, \boldsymbol{R} >$, where $\boldsymbol{S}$ is a finite set of N states; $\boldsymbol{A}$ is a set of M actions that the learner can take in a given state; $\mathcal{P}_{sa} : \boldsymbol{S} \times \boldsymbol{A} \times \boldsymbol{S} \rightarrow [0, 1]$ describes the transition dynamics of the environment, i.e., $\mathcal{P}_{sa} \triangleq \mathcal{P}(s', a, s)$ is the probability of transitioning to state s' after taking action a in state s; the initial state s_0 is drawn from the initial state distribution $\boldsymbol{\alpha}$ with $\sum_s \alpha_s = 1$; $\gamma \in \mathbb{R} \rightarrow [0, 1)$ is the discount factor to control the effect of rewards obtained in future; r_s is the reward

perceived in a given state s. Rewards are obtained by a linear combination of a set of known features, $r_s = \boldsymbol{w}^\top \boldsymbol{\phi_s}$, where $\boldsymbol{\phi}_s : S \to \mathbb{R}^F$ denote the mapping from state s to a set of F task-dependent features; $\boldsymbol{w} \in \mathbb{R}^F_{[-1,1]}$ and $\|\boldsymbol{w}\|_1 \leq 1$ define the relative weights of the features. Different weights for the features yield different rewards while interacting with the environment.

A policy $\pi \in \Pi$ defines the mapping from state to actions. A policy can be deterministic, $\pi_s : S \to A$, in which case each state is mapped to a unique action, or a policy can be stochastic in which case each state is mapped to a distribution over actions, $\pi_{sa} : S \times A \to [0,1]$ and $\sum_a \pi_{sa} = 1$. A stochastic policy represented as a convex combination of optimal deterministic policies is called a mixed policy. A mixed policy is executed by selecting the optimal deterministic policy π_i at $t = 0$ in the set with probability $\lambda_i (\lambda_i \geq 0, \sum_i \lambda_i = 1)$, and following it for the rest of the time.

The *value function* $V_s^\pi : S \to \mathbb{R}$ measures the expected value of discounted sum of rewards that the agent gains starting from state s and following policy π,

$$V_s^\pi = E\Big\{ \sum_{t=0}^{T} \gamma^t r_{s_t} | s_t = s, a = \pi_{s_t} \Big\}, \tag{2.1}$$

where, $s_{t+1} \sim \mathcal{P}^\pi(s_t)$ and $\mathcal{P}^\pi : S \times S \to [0,1]$, is the transition dynamics after fixing action in each state according to policy π, T is the horizon or range of the sequence of states and actions mapped in future given the current state. When starting from the initial state distribution α_s, the value of a policy π reduces to a scalar defined by: $V^\pi = \sum_s \alpha_s V_s^\pi$ (note that we dropped the s in the subscript). The quality of a policy may also be evaluated using the *action-value function* $Q_{s,a}^\pi : S \to \mathbb{R}$ in case the transition dynamics are unknown,

$$Q_{s,a}^\pi = E\Big\{ \sum_{t=0}^{T} \gamma^t r_{s_t} | s_t = s, a_t = a, a = \pi_{s_t} \Big\}. \tag{2.2}$$

The goal in RL is to adjust the policy vector such that the future discounted rewards received on average during its course of actions are maximized. A policy π is optimal for the MDP if it satisfies

$$\pi = \arg\max_{\pi \in \Pi} V^\pi. \tag{2.3}$$

Similar to how the value-function gives an expectation over rewards in the long run, *feature expectation* vector, $\boldsymbol{\mu}_s^\pi : S \to \mathbb{R}^f$, corresponds to the discounted sum of the features as the agent observes the sequence $s_0, s_1, \ldots, s_T$ starting from the state $s_0 = s$ following policy π, i.e., $\boldsymbol{\mu}_s^\pi = E\{\sum_{t=0}^{T} \gamma^t \boldsymbol{\phi}_{s_t} | s_0 = s, a = \pi(s_t)\}$. Note that the reward function is linear in features, the value-function is also linear in feature expectations, parametrized by the same weight vector $\boldsymbol{w}$, i.e., $V_s^\pi = \boldsymbol{w}^\top \boldsymbol{\mu}_s^\pi$ and similarly for the initial state distribution,

$V^{\pi} = \boldsymbol{w}^{\top}\boldsymbol{\mu}^{\pi}$, where $\boldsymbol{\mu}^{\pi} = \sum_s \alpha_s \boldsymbol{\mu}_s^{\pi}$.

The demonstrations performed by the human/expert is represented by its feature expectation vector $\boldsymbol{\mu}^{\pi_E}$. Given the sequence of visited states $\mathcal{D}^{\pi_E} = \{(s_0^{(i)}, a_0^{(i)}), (s_1^{(i)}, a_1^{(i)}), \ldots, (s_T^{(i)}, a_T^{(i)})\}_{i=1}^{M}$ and the corresponding features over M demonstrations $\{\boldsymbol{\phi}_{s_0}^{(i)}, \boldsymbol{\phi}_{s_1}^{(i)}, \ldots, \boldsymbol{\phi}_{s_T}^{(i)}\}_{i=1}^{M}$, an empirical estimate of the feature expectation of the demonstrator can be computed as

$$\hat{\boldsymbol{\mu}}^{\pi_E} = \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{T}\gamma^t \boldsymbol{\phi}_{s_t}^{(i)}. \tag{2.4}$$

The goal of IRL is to find the reward function parameters $\boldsymbol{w}$ such that the resulting optimal policy of the learner π_L yields the same rewards or value as observed in the demonstrations (assuming the demonstrations to be optimal), i.e., $|V^{\pi_E} - V^{\pi_L}| < \varepsilon_1$, where ε_1 is a small positive number. The demonstrations and the trajectory samples obtained from the transition dynamics after fixing the learner policy π_L are compared to evaluate the estimated reward function.

The first treatment of IRL used a **linear program** to recover the unknown reward function under the constraint that the demonstrations are drawn from some optimal policy [Ng 2000]. Abbeel and Ng in [Abbeel 2004] gave formal guarantees required to match the performance of the learner with that of the demonstrator measured in terms of policy values. They present two algorithms based on the idea of **matching feature-expectation** vectors with same performance guarantees. The feature-expectation matching algorithms return the learner's policy π_L for a given expert strategy such that $\|\boldsymbol{\mu}^{\pi_E} - \boldsymbol{\mu}^{\pi_L}\|_2 \leq \varepsilon_1$, thereby yielding the same performance as that of the demonstrator,

$$\begin{aligned} |V^{\pi_E} - V^{\pi_L}| &= \boldsymbol{w}^{\top}(\boldsymbol{\mu}^{\pi_E} - \boldsymbol{\mu}^{\pi_L}) \\ &\leq \|\boldsymbol{w}\|_2 \|\boldsymbol{\mu}^{\pi_E} - \boldsymbol{\mu}^{\pi_L}\|_2 \\ &\leq 1 \cdot \varepsilon_1, \end{aligned} \tag{2.5}$$

where the first inequality follows from Cauchy-Schwarz inequality: $|\boldsymbol{x}^{\top}\boldsymbol{y}| \leq \|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2$. The expression shows that IRL can be simplified to finding that optimal policy for some reward function whose feature expectation is same as that of the demonstrator. Syed and Schapire in [Syed 2008] proposed a **game-theoretic approach** to find a policy (and the corresponding reward function) on the pareto optimal curve that performs at least as well as the demonstrator, thereby, allowing imperfect demonstrations. Howard *et al.* used this approach for transferring impedance from human to a robot arm [Howard 2010]. Ratliff *et al.* introduced the **max-margin formulation** under which the demonstrated policy

samples perform better than all other policies by a margin [Ratliff 2006]. The approach was extended to LEARCH – LEArning and seaRCH – in order to handle non-linear reward functions [Ratliff 2009]. Ramachandran and Amir in [Ramachandran 2007] presented the **Bayesian formulation** to maximize the posterior probability of the reward function given the observation sequences parametrized by the reward function (see also [Neu 2012] and [Mombaur 2010] for IRL as parameter estimation problem). Similar to Bayesian formulation of IRL, policies with higher rewards are exponentially more favoured in the **maximum entropy** formulation and equivalently, policies with equivalent rewards have equal probabilities [Ziebart 2008], i.e.,

$$\mathcal{P}(\mathcal{D}^{\pi_E}|w) = \prod_{i=1}^{M} \frac{1}{Z^{\pi}} e^{V^{\pi_E^{(i)}}} = \prod_{i=1}^{M} \frac{e^{V^{\pi_E^{(i)}}}}{\int e^{V^{\pi}} d\pi} = \prod_{i=1}^{M} \frac{e^{\boldsymbol{w}^{\top}\boldsymbol{\mu}^{\pi_E^{(i)}}}}{\int e^{\boldsymbol{w}^{\top}\boldsymbol{\mu}^{\pi}} d\pi}, \tag{2.6}$$

where the denominator of this distribution is called the *partition function* and it becomes computationally intractable for even moderately high-dimensional spaces. The optimal value of the reward function parameters $\boldsymbol{w}$ can be obtained by maximizing the log-likelihood of the observed demonstrations in the maximum entropy formulation, $\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} \log \mathcal{P}(\mathcal{D}^{\pi_E}|\boldsymbol{w})$. The maximum entropy formulation gained a lot of popularity after its formulation, and a number of bottlenecks of IRL have been addressed using this formulation.

Thus far, we made a number of assumptions in our formulation of addressed IRL algorithms, including: 1) the intention (underlying reward function) is same in all the observed demonstrations, 2) there is an oracle that yields the optimal policy samples for each candidate reward function in an inner loop, and 3) the transition dynamics of the environment is known.

In our work presented below [Tanwani 2013b], we relax the limitation of having the same reward function in all the demonstrations. Note that learning the underlying reward function from demonstrations is an ill-posed problem as many feature combinations yield the same optimal policy. For example, setting $\boldsymbol{w} = 0$ would yield the same value irrespective of the underlying policy. This often leads to careful engineering of the features to get a meaningful reward function and the resulting optimal policy. On the contrary, we are interested in learning multiple ways of performing a task by observing several experts' demonstrations that are driven by different reward functions. A naive way of learning each reward function separately would significantly increase the computational overhead of finding optimal policies. To circumvent this computational burden, we exploit the fact that all the demonstrations share the same transition dynamics and only differ in the underlying reward function. This allows us to transfer the learned experience and bootstrap

incremental learning of multiple policies.

2.2 Learning Reward Function(s) in Discrete Domains

It is well-known that humans vary widely in performing sequential decision-making tasks, possibly differing in their intentions or ways of gauging task-dependent features. This difference is a fundamental trait of natural selection that contributes to fitness and survival of an individual in changing environments. Consequently, there are often several useful ways of performing a task and how one assesses multiple criteria in a given situation yields the goodness of a decision. For example, a demonstrator may have preference to drive fast, overtake other cars and stay in left lane, while other demonstrator may want to drive slow and keep safe distance from other cars. In the case of throwing a ball to another person, the demonstrator may want to throw the ball very high, while other demonstrator may want to roll the ball along the floor. Despite these scenarios, most of the previous work in IRL assumes a single demonstrator having the same intention in all the demonstrations – albeit with a few exceptions. In [Babes 2011], the authors use an expectation-maximization approach to cluster similar strategies in the demonstrations where the number of clusters defined a priori represent the number of reward functions. Dimitrakakis and Rothkopf [Dimitrakakis 2012] present a Bayesian approach to learn multiple reward functions by considering joint prior on reward functions and policies. Choi and Kim in [Choi 2012] present a non-parametric Bayesian approach using the Dirichlet process mixture model to learn multiple reward functions. In contrast to the above work, we take a direct geometric approach to learn a convex set of optimal policies enclosing all the demonstrations. This allows us to efficiently match any previously unseen expert strategy drawn from this set. Moreover, our method of learning multiple strategies is incremental and allows transfer of knowledge; contrary to all the aforementioned batch learning approaches for multiple reward functions. In this section, we first formalize our problem statement and present our multiple reward functions learning approach and then explain the transfer of knowledge to speed up the learning process.

Let Π_D be the set of all deterministic stationary policies available to the robot/learner in a MDP as possible ways of executing a task. Each policy possibly gives a different feature expectation $\boldsymbol{\mu}^{\pi}$, among which the optimal ones maximize the value of a policy V^{π} for some reward function $\boldsymbol{w}$. The set of feature expectations $\boldsymbol{\mu}^{\pi_1}, \boldsymbol{\mu}^{\pi_2}, \ldots, \boldsymbol{\mu}^{\pi_d} \subseteq \boldsymbol{\mu}(\Pi_D)$ that are maximal for some $\boldsymbol{w}$ defines a convex hull $\mathrm{Co}\{\boldsymbol{\mu}(\Pi_D)\}$ in the feature expectation space. Ideally, we would like to learn all the optimal policies over this convex hull so that the learner can readily replicate any demonstrator by appropriately choosing

among the optimal deterministic policies for the corresponding reward function. Note that the deterministic stationary policies of Π_D alone do not constitute all the feasible strategies in the feature expectation space. For example, if the expert is sub-optimal, the feature expectation vector lies within the convex hull of optimal deterministic policies. Alternatively, if an expert performs the demonstrations optimally with respect to different reward functions (first episode with some reward function, and second episode with another reward function), the feature expectation vector also lies within the convex hull of optimal deterministic policies. Here, we assume that the demonstrations are always optimal either in a deterministic or a stochastic manner. We do not limit an expert to be optimal or nearly-optimal in a deterministic way; otherwise we could select one optimal deterministic policy with feature expectation $\boldsymbol{\mu}^{\pi_i}$ lying on the convex hull that is closest to $\boldsymbol{\mu}^{\pi_E}$. We only require the demonstrations of an expert to lie within the convex hull of feature expectations, and approximate such an expert strategy with a mixture of optimal policies.

However, learning all optimal policies in Π_D is in general intractable with its cardinality $|\Pi_D| = \boldsymbol{A}^{\boldsymbol{S}}$. Moreover, not all the policies in the set lead to practically useful description of a task. For example, in the case of the ball throwing example, there will be a range of parameter values of $\boldsymbol{w}$ for which the ball may not even reach the user or the car may hit other cars. We rely upon the experts' demonstrations to learn useful policies only in a tractable manner. Let us denote Π_E as the set of deterministic policies of the demonstrators where $|\Pi_E| \ll |\Pi_D|$ in general. Let $\Delta(\Pi_E)$ be the set of probability distributions (unknown) over the set Π_E from which the demonstrators draw a finite number of strategies $\boldsymbol{\mu}^{\pi_{E_1}}, \boldsymbol{\mu}^{\pi_{E_2}}, \ldots, \boldsymbol{\mu}^{\pi_{E_n}}$ as possible useful ways of demonstrating a task to the learner. The goal of the learner is to approximate the demonstrated strategies as $\boldsymbol{\mu}^{\pi_{A_1}}, \boldsymbol{\mu}^{\pi_{A_2}}, \ldots, \boldsymbol{\mu}^{\pi_{A_n}}$ belonging to the probability distribution set $\Delta(\Pi_A)$, where Π_A contains the optimal deterministic policies of the learner $\{\pi_1, \pi_2, \ldots, \pi_T\} \in \Pi_A$ corresponding to the reward functions $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_T\}$. After experiencing a finite number of demonstrators, the learner should be able to approximate any new demonstrated strategy drawn from $\Delta(\Pi_E)$[1]. The learner does so by finding the set of deterministic policies Π_A that is used to generate a mixed policy for matching any demonstrated strategy by drawing from the associated distribution such that the performance of the learner is at least as good as that of the expert with a tolerance of ε_0:

$$|V^{\pi_E} - V^{\pi_A}| \leq \varepsilon_0, \tag{2.7}$$

where $\varepsilon_0 \geq 0$, $\pi_A \sim \Delta(\Pi_A)$, $\pi_E \sim \Delta(\Pi_E)$ and the demonstrator's reward function (weight vector) is unknown in the demonstrated strategy.

[1]For testing, we draw the new demonstrator strategy by convex combination of already experienced strategies $\boldsymbol{\mu}^{\pi_{E_1}}, \boldsymbol{\mu}^{\pi_{E_2}}, \ldots, \boldsymbol{\mu}^{\pi_{E_n}}$.

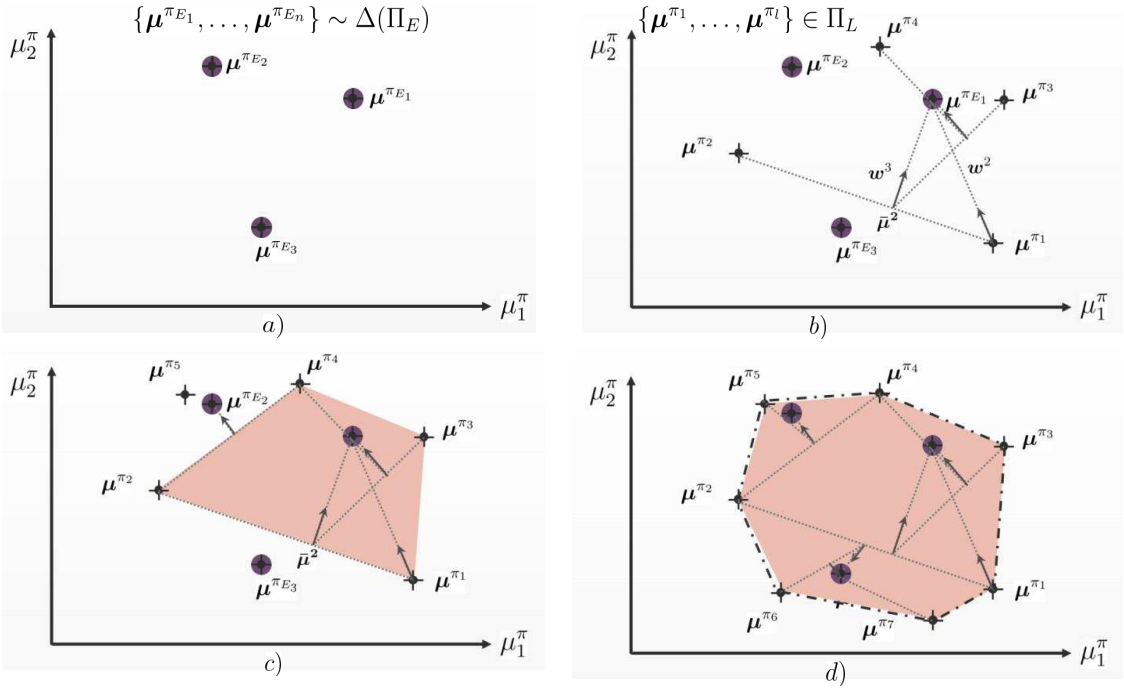


Figure 2.1: Learning multiple reward functions with IRL using the projection algorithm: *(top-left)* feature expectations of expert strategies, *(top-right)* optimal policies learned for first expert strategy, *(bottom-left)* incremental learning of next expert strategy, *(bottom-right)* convex set of optimal policies enclosing all expert strategies.

2.2.1 Multiple Reward Functions

Our problem formulation applies to all the feature matching approaches described above for learning multiple reward functions. Here, we build upon the projection algorithm [Abbeel 2004] for learning multiple reward functions. The projection algorithm works by iteratively finding an optimal policy π_i for the reward function, $r_s^i = \boldsymbol{w}_i^\top \boldsymbol{\phi}_s$, in each iteration $i = 1 \dots T$, starting from some randomly chosen reward function parameters. The reward function $\boldsymbol{w}_i$ is updated by a projection mapping $\bar{\boldsymbol{\mu}}_i$ that gradually reduces the norm of the weight vector until the weight vector changes no more and the learning converges (see Algorithm 1). At the end, the point $\boldsymbol{\mu}^{\pi_E}$ is guaranteed to be close to the convex hull of the feature expectation set of intermediate policies, $\boldsymbol{\mu}^{\pi_1}, \boldsymbol{\mu}^{\pi_2}, \dots, \boldsymbol{\mu}^{\pi_T}$, with $\boldsymbol{\mu}^{\pi_A}$ being the closest point in that convex hull to $\boldsymbol{\mu}^{\pi_E}$. Mixed policy $\boldsymbol{\mu}^{\pi_A}$ is generated by a convex combination of intermediate policies. It can be shown that the mixed policy $\boldsymbol{\mu}^{\pi_A}$ which performs approximately as good as the demonstrator can be generated in $O(T \log T)$ iterations.

After computing the feature expectation set $\boldsymbol{\mu}^{\pi_1}, \boldsymbol{\mu}^{\pi_2}, \dots, \boldsymbol{\mu}^{\pi_T}$ corresponding to T iterations of the projection algorithm for the demonstrated strategy $\boldsymbol{\mu}^{\pi_{E_1}}$, the initial weight vector for $\boldsymbol{\mu}^{\pi_{E_2}}$ is selected along the line connecting $\boldsymbol{\mu}^{\pi_{E_2}}$ and the closest possible feature expectation achievable from the set $\boldsymbol{\mu}^{\pi_1}, \boldsymbol{\mu}^{\pi_2}, \dots, \boldsymbol{\mu}^{\pi_T}$ to $\boldsymbol{\mu}^{\pi_{E_2}}$. For the j-th demonstrator, the initial weight is computed as: $\boldsymbol{w} = \boldsymbol{\mu}^{\pi_{E_j}} - \boldsymbol{u}$, where $\boldsymbol{u}$ is obtained from the feature-expectation set as

$$\begin{aligned} &\min_{\boldsymbol{\mu}} \|\boldsymbol{\mu} - \boldsymbol{\mu}^{\pi_{E_j}}\|_2 \qquad \text{s.t.} \\ &\boldsymbol{\mu} = \textstyle\sum_{i=1}^{(T\times j)} \lambda_i \boldsymbol{\mu}^{\pi_i}, \qquad \textstyle\sum_{i=1}^{(T\times j)} \lambda_i = 1, \qquad \lambda_i \geq 0. \end{aligned} \tag{2.8}$$

Note that if $\|\boldsymbol{w}\|_2 < \varepsilon_1$ after the above optimization, the algorithm terminates in the first iteration as $\boldsymbol{\mu}^{\pi_{E_j}}$ can already be estimated from the existing feature expectation set of the learner.

2.2.2 Transfer Learning in Optimal Policy Search

There are two main issues in learning multiple reward functions with the feature-matching approach: 1) it is computationally expensive to find an optimal policy for a given reward function with weight $\boldsymbol{w}$, and 2) the number of deterministic policies in the set Π_A can grow arbitrarily large for matching all the demonstrated strategies. Consequently, the learner seeks to: 1) reuse the previously learned policies to achieve faster learning with the new reward function parameters $\boldsymbol{w}$, and 2) store only distinct policies (we call them ε-better

policies) that are possibly optimal for a wide range of weights.

Let $\Pi_A^{(j)}$ be the set of stored optimal deterministic policies after learning the j-th demonstrated strategy. Given a new reward function with weight $\boldsymbol{w}$, the learner chooses as initial policy π_{init} the one with the highest value in the set $\Pi_A^{(j)}$,

$$\pi_{init} = \arg \max_{\pi \in \Pi_A^{(j)}} (\boldsymbol{w}^\top \boldsymbol{\mu}^\pi). \tag{2.9}$$

The initial policy π_{init} is considered as the optimal policy for the new reward function $\boldsymbol{w}$ if there exists no other policy whose performance is ε-better than the initial policy. The set of ε-better policies is characterized in the following Lemma:

Lemma 1 *Given a finite state space $\boldsymbol{S}$, action set $\boldsymbol{A}$, initial state distribution $\boldsymbol{\alpha}$, reward function $\boldsymbol{R}$, the optimal policy π with transition matrix $\mathcal{P}^\pi$ is ε-better than an initial policy π_{init} with transition matrix $\mathcal{P}^{\pi_{init}}$, if it satisfies,*

$$\boldsymbol{\alpha}^\top \left((I - \gamma \mathcal{P}^\pi)^{-1} - (I - \gamma \mathcal{P}^{\pi_{init}})^{-1} \right) \boldsymbol{R} \geq \varepsilon. \tag{2.10}$$

For proof, see [Tanwani 2013b]. Lemma 1 gives the space of policies that are better than π_{init} for the given reward function with weight $\boldsymbol{w}$. We now further narrow down this space by imposing constraints due to other policies in the set $\Pi_A^{(j)}$.

Definition 1 *Given a set of optimal deterministic policies, $\pi_1, \pi_2, \ldots, \pi_T \in \Pi_A$, with feature expectations, $\boldsymbol{\mu}^{\pi_1}, \boldsymbol{\mu}^{\pi_2}, \ldots, \boldsymbol{\mu}^{\pi_T} \in \boldsymbol{\mu}(\Pi_A)$, corresponding to reward functions with weights, $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_T$, the optimal policy π for reward function with weight $\boldsymbol{w}$ and feature expectation $\boldsymbol{\mu}^\pi$ is an ε-better policy in Π_A if,*

$$\boldsymbol{w}^\top (\boldsymbol{\mu}^\pi - \boldsymbol{\mu}^{\pi_i}) \geq \varepsilon \tag{2.11}$$

$$(\boldsymbol{w}_i)^\top (\boldsymbol{\mu}^\pi - \boldsymbol{\mu}^{\pi_i}) \leq 0 \qquad i = 1, 2, \ldots, T. \tag{2.12}$$

Adding constraints (2.11) and (2.12) and using Cauchy-Schwarz inequality gives a lower bound on the distance between $\boldsymbol{w}$ and other weight vectors in the set $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_T$ for

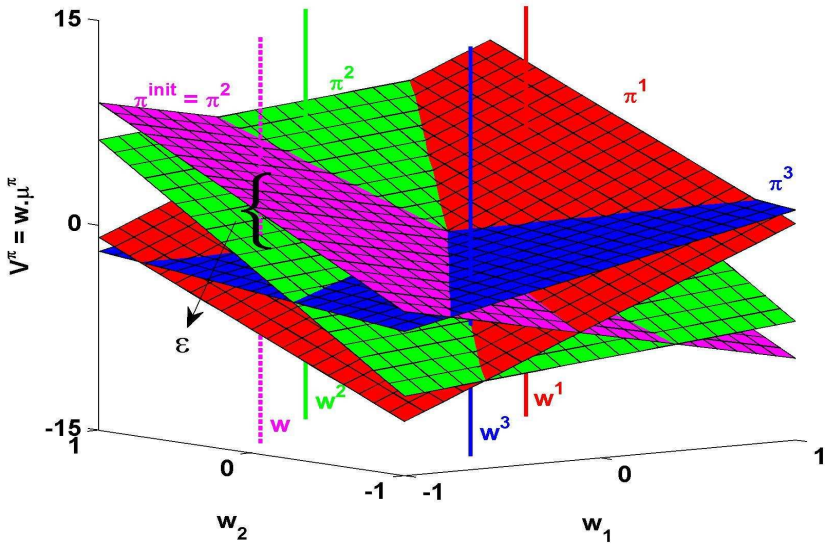


Figure 2.2: 'Value-Surface' with $f = 2$ (best viewed in color). For a new reward function with weight $\boldsymbol{w}$, value-surface gives the initial policy with the best weighted value. The surface is updated only if there exists a ε-better policy at $\boldsymbol{w}$ whose weighted value is less than the value of other optimal policies at $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_T$.

$\boldsymbol{w}$ to have an ε-better policy[2]:

$$\begin{aligned} (\boldsymbol{w} - \boldsymbol{w}_i)^T(\boldsymbol{\mu}^\pi - \mu^{\pi_i}) &\geq \varepsilon \\ \|\boldsymbol{w} - \boldsymbol{w}_i\|_2 \, \|\boldsymbol{\mu}^\pi - \boldsymbol{\mu}^{\pi_i}\|_2 &\geq \varepsilon \\ \|\boldsymbol{w} - \boldsymbol{w}_i\|_2 &\geq \frac{\varepsilon(1-\gamma)}{\sqrt{F}} \qquad i = 1 \ldots T. \end{aligned} \tag{2.13}$$

Every policy adds a set of constraints for a new reward function with weight $\boldsymbol{w}$ to satisfy. The set $\boldsymbol{\mu}^{\pi_1}, \boldsymbol{\mu}^{\pi_2}, \ldots, \boldsymbol{\mu}^{\pi_T}$ defines a convex hull $\text{Co}\{\boldsymbol{\mu}(\Pi_A)\}$ in the feature expectation space and the resulting piecewise planar *value-surface* gives the best policy value for each possible weight (see Fig. 2.2). Given that $\|w\| \leq 1$, this elicits an upper bound on $\varepsilon \leq \frac{2\sqrt{F}}{1-\gamma}$.

Note that Lemma 1 combined with the constraints in Definition 1 can be used to find an ε-better policy with a linear program; albeit the computation may be slow. In our implementation, we verify the existence of ε-better policy in three steps as follows: 1) satisfy (2.13) to check if there does not exist any $\boldsymbol{w}_i$ in the vicinity of $\boldsymbol{w}$ for which we already have the optimal policy, 2) there exists a $\boldsymbol{\mu}$ such that the constraints in Definition

[2]Remember that: $\boldsymbol{\phi} \in \mathbb{R}^F_{[0,1]} \Rightarrow \boldsymbol{\mu}^\pi \in \mathbb{R}^F_{[0,\frac{1}{1-\gamma}]} \Rightarrow \|\boldsymbol{\mu}^\pi - \boldsymbol{\mu}^{\pi_i}\|_2 \leq \frac{\sqrt{F}}{1-\gamma}$.

1 are satisfied, i.e.,

$$\begin{aligned} \text{Solve for } \boldsymbol{\mu} \text{ s.t.} \quad & \boldsymbol{w}^\top(\boldsymbol{\mu} - \boldsymbol{\mu}^{\pi_{init}}) \geq \varepsilon, \\ & (\boldsymbol{w}_i)^\top(\boldsymbol{\mu} - \boldsymbol{\mu}^{\pi_i}) \leq 0, \qquad i = 1, 2, \ldots, T \\ & 0 \preceq \boldsymbol{\mu} \preceq \tfrac{1}{1-\gamma}\boldsymbol{I}. \end{aligned} \tag{2.14}$$

Note that the use of $\boldsymbol{\mu}^{\pi_{init}}$ at $\boldsymbol{w}$ also satisfies all $\boldsymbol{\mu}^{\pi_i}$ in (2.11), and 3) find the optimal policy using the value-iteration algorithm starting from π_{init} (see Sec. 2.3 for use of other RL algorithms). If the verification fails at any of the above three steps, π_{init} is declared the optimal policy for $\boldsymbol{w}$. The overall algorithm of learning multiple reward functions from demonstrations is presented in Algorithm 1.

2.2.3 Grid World and Mini-Golf Examples

2.2.3.1 Grid World Example

Let's consider a simple grid world environment of 100×100 cells, where each cell represents a different state of the learner. In a given state, the learner can take 9 different actions corresponding to a move in all eight neighbouring directions or a stay in the same cell. Transition dynamics are stochastic with 0.7 probability of moving in the direction of desired action instead of a random one. Initial state distribution is uniform over all the states. Five features – radial basis functions with centres chosen randomly among states and width drawn in the interval $[1, 20]$ – are used to populate the feature space. Ten different reward functions are generated to simulate multiple demonstrators by randomly assigning different weights to every feature in the interval $[-1, 1]$. We log the visited states sequence of 125 time steps from the optimal policy of every reward function in a demonstration and vary the number of sample demonstrations to study its effect on learning multiple reward functions.

Experimental study is performed on a grid world task to assess the performance of optimal policy transfer in learning multiple reward functions with different values of ε against the 'no transfer' case where each demonstrated strategy is learned separately with the projection algorithm. The performance is evaluated using three metrics: 1) empirical error – distance between the estimated feature expectation of the demonstrator and the learner averaged over n strategies, i.e., $\frac{1}{n}\sum_{j=1}^{n}\|\hat{\boldsymbol{\mu}}^{\pi_{E_j}} - \hat{\boldsymbol{\mu}}^{\pi_{A_j}}\|_2$, 2) CPU learning time, and 3) number of policies stored. We use the same discount factor of 0.9 in all our experiments. Moreover, we only iterate our algorithm for a demonstrated strategy up to a maximum of 50 iterations.

Algorithm 1 *Transfer in IRL for Multiple Reward Functions*

Input: $< \boldsymbol{S}, \boldsymbol{A}, \mathcal{P}_{sa}, \boldsymbol{\alpha}, \gamma, \phi, \{\boldsymbol{\mu}^{\pi_{E_1}}, \boldsymbol{\mu}^{\pi_{E_2}}, \ldots, \boldsymbol{\mu}^{\pi_{E_n}}\}, \varepsilon >$

procedure `LEARNER_TRAINING`

Initialize $i := 1$, $\boldsymbol{w}_i$ s.t. $\|\boldsymbol{w}_i\|_1 = 1$, $\Pi_A = \{\}$
$\bar{\boldsymbol{\mu}}^i = \arg\max_{\boldsymbol{\mu} \in \boldsymbol{\mu}(\Pi_D)} ((\boldsymbol{w}_i)^\top \boldsymbol{\mu})$
for $j = 1$ **to** $|\boldsymbol{\mu}^{\pi_{E_n}}|$ **do**
  if $\Pi_A \neq \{\}$ **then**
    Solve (2.8) for $\boldsymbol{\mu} := \min_{\boldsymbol{\mu} \in \mathrm{Co}\{\boldsymbol{\mu}(\Pi_A)\}} \|\boldsymbol{\mu} - \boldsymbol{\mu}^{\pi_{E_j}}\|_2$
    $\boldsymbol{w}_i = \boldsymbol{\mu}^{\pi_{E_j}} - \boldsymbol{\mu}$
    $\bar{\boldsymbol{\mu}}_{i-1} = \boldsymbol{\mu}$
  end if
  repeat
    if $i > 1$ **then**
      $\pi^{init} := \arg\max_{\pi \in \Pi_A} ((\boldsymbol{w}_i)^\top \boldsymbol{\mu}^\pi)$
      Verify three steps for existence of ε-better policy
      if three steps are verified **then**
        Add π_i to Π_A
      else
        $\pi_i = \pi_{init}$
      end if
      $\bar{\boldsymbol{\mu}}_i = \bar{\boldsymbol{\mu}}_{i-1} + \frac{(\boldsymbol{\mu}^{\pi_i} - \bar{\boldsymbol{\mu}}_{i-1})^\top (\boldsymbol{\mu}^{\pi_{E_j}} - \bar{\boldsymbol{\mu}}_{i-1})}{(\boldsymbol{\mu}^{\pi_i} - \bar{\boldsymbol{\mu}}_{i-1})^\top (\boldsymbol{\mu}^{\pi_i} - \bar{\boldsymbol{\mu}}_{i-1})} (\boldsymbol{\mu}^{\pi_i} - \bar{\boldsymbol{\mu}}_{i-1})$
    end if
    $\boldsymbol{w}_{i+1} = \boldsymbol{\mu}^{\pi_{Ej}} - \bar{\boldsymbol{\mu}}_i$
    $i := i + 1$
  until $\|\boldsymbol{w}_i - \boldsymbol{w}_{i-1}\|_2$ is unchanged
end for
return set of learner policies Π_A

procedure `LEARNER_TESTING`

loop
  Demonstration of a strategy $\boldsymbol{\mu}^{\pi_E} \sim \Delta(\Pi_E)$
  Learner finds a strategy $\boldsymbol{\mu}^{\pi_A} \sim \Delta(\Pi_A) : \boldsymbol{\mu}^{\pi_A} = \sum_{i=1}^{|\Pi_A|} \lambda_i \boldsymbol{\mu}^{\pi_i}$, where λ_i is obtained by solving (2.8) with $(T \times j) = |\Pi_A|$
end loop

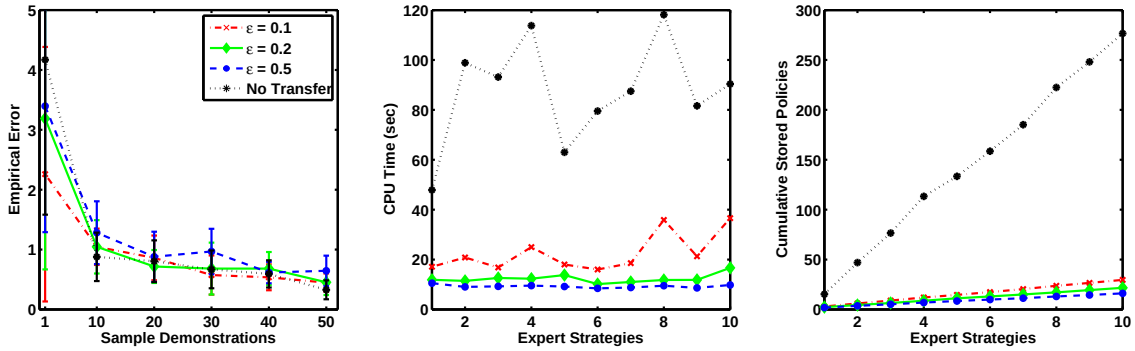


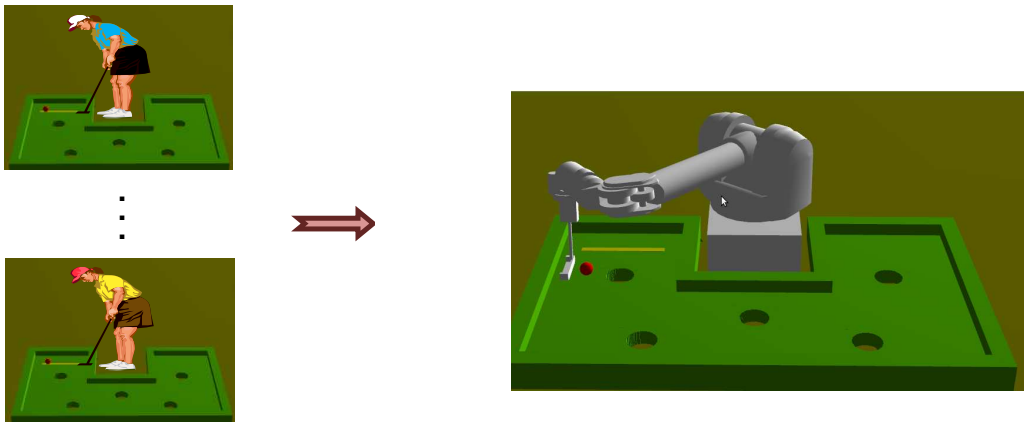
Figure 2.3: Grid world results. Results are averaged over 5 iterations.

Fig. 2.3 (left) shows that the average empirical error over all strategies decreases sharply with the increase in the number of samples in a demonstration, while it increases slightly with higher values of ε for a given number of sample demonstrations. The other two plots clearly indicate the advantage of optimal policy transfer with a clear performance improvement in terms of required time and number of policies to learn all strategies. Note that the optimal policy transfer is useful even for the case of learning a single expert strategy.

2.2.3.2 Mini-Golf Example

The goal in mini-golf, short for Miniature golf, is to sink the ball into the hole from the tee area in as few shots as possible. The simulated mini-golf environment is shown in Fig. 2.4. To simulate various strategies of demonstrations, there are 5 holes in the field for 5 demonstrators each having preference to sink the ball in a different hole. The learner is required to estimate the set of deterministic policies Π_A from which it can approximate any randomly chosen distribution over the demonstrated strategies. In other words, sink the ball in each hole same number of times as the demonstrator does in his/her strategy.

State, Action and Feature Space: The state-space corresponds to the 2−dimensional position of the ball in the grid, $|S|= 81 \times 56 = 4536$. The action-set corresponds to 4 hitting directions at right angles to one another and 6 different hitting speeds, $|A|= 24$. The feature space is 13-dimensional, where first 8-dimensions give distance of the ball to each wall segment, and other 5-dimensions give distance of the ball to each hole. The features are scaled such that $\phi(s) \leq 1$. Intuitively speaking, an ideal strategy chooses the intermediate ball positions in a way that keeps the ball maximally away from all other holes and wall segments, and sinks the ball in the desired hole in least number of shots. The initial state distribution is uniform on the tee area marked with the yellow line in Fig.



$\{\boldsymbol{\mu}^{\pi_{E_1}}, \ldots, \boldsymbol{\mu}^{\pi_{E_n}}\} \sim \Delta(\Pi_E)$

Figure 2.4: Simulated mini-golf playing field with different expert strategies on left and the Barrett WAM robot arm learning to play mini-golf on right.

2.4. An episode of play corresponds to 50 shots. The ball position is randomly reset on the tee area every time the episode ends or the ball sinks into a hole.

Results and Discussions: We design our experiments such that the learner is required to approximate the 5 demonstrated strategies from their estimated feature expectations using our proposed approach in the training phase. During testing, we draw 50 mixed strategies each corresponding to a random distribution over pure demonstrated strategies, and the learner is asked to replicate the demonstrated strategy. Fig. 2.5 gives a measure of the ability of the learner to replicate previously unseen demonstrated strategies. It is seen that after learning the 5 demonstrated strategies corresponding to sinking the ball in each hole separately during training, the learner is able to successfully replicate all the mixed strategies in the testing phase.

IRL has received a lot of attention in recent times, but its applications have often been limited to discrete domains. The situation in continuous spaces is rather different. We first visit the problem of finding optimal policy in continuous spaces and subsequently, find reward function in continuous unknown environments.

2.3 Reinforcement Learning in Continuous Domains

Rewards-driven learning from demonstrations in continuous domains becomes significantly challenging with approximation methods required to estimate transition dynamics, reward function, and the optimal policy. Applying RL/IRL on real-world control problems is diffi-

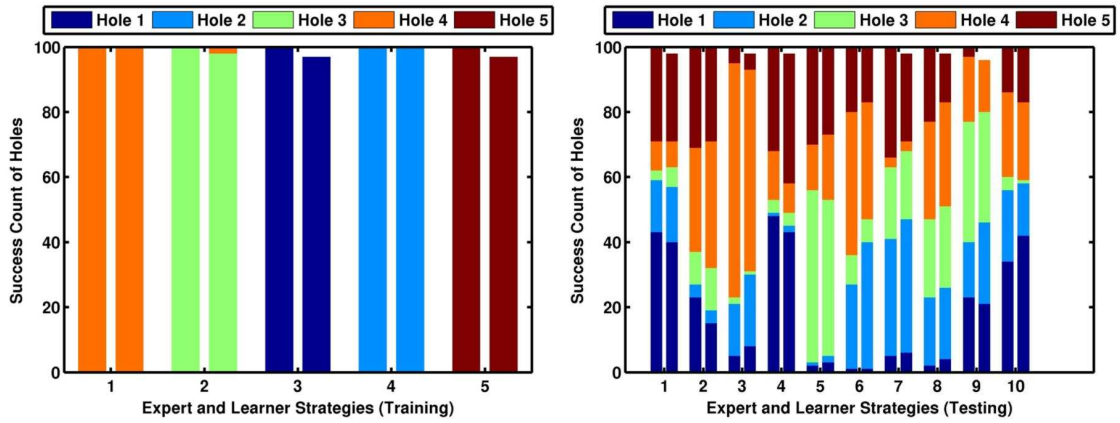


Figure 2.5: Comparison of first 15 expert and learner strategies for 100 episodes with $\varepsilon = 0.1$. For every strategy number, the first bar gives the success count of holes for the expert, the second bar gives the learner's response to the expert's strategy. First five strategies on left correspond to the training set, other mixed strategies are from the testing set on the right. Holes are numbered from left to right in Fig. 2.4.

cult for a number of reasons. First, the policy search becomes computationally intractable for moderately high dimensional spaces. Second, it takes too many samples of interaction with the environment to obtain the optimal policy, leading to exploration vs exploitation problem. Third, a lot of time is elapsed in constructing a single reward function from several desired objectives the learner is expected to optimize for in the policy. Finally, the optimal policy is sensitive to modelling changes in the environment and does not scale across related tasks/environmental situations easily.

Let us denote $\boldsymbol{\xi}_t \in \mathbb{R}^D$ for the state in continuous domain, $\boldsymbol{u}_t \in \mathbb{R}^m$ for the action or control input, and $\boldsymbol{\xi}_{t+1} \sim \mathcal{P}(\boldsymbol{\xi}_t, \boldsymbol{u}_t)$ to represent the stochastic environment under the MDP framework. When the environment is deterministic, we describe the transition dynamics with a non-linear function, $\boldsymbol{\xi}_{t+1} = f(\boldsymbol{\xi}_t, \boldsymbol{u}_t)$. The action $\boldsymbol{u}_t$ is generated by the policy π representing the family of probability distributions $\boldsymbol{u}_t \sim \pi(\boldsymbol{\xi}_t; \theta)$, where θ represents the policy parameters. The learner aims to find a distribution that maximizes the probability of sampling those actions which yield higher rewards. The learning control methods are classified as [Sutton 1992]: 1) *indirect/model-based* – optimal control policy is recomputed from an estimated model of the environmental dynamics at each iteration (without any explicit exploration noise), and 2) *direct/model-free* – optimal control policy is determined without any model of the transition dynamics of the environment. Noise is added to the policy parameters during learning to search for optimal actions in a given state.

There exists a large repertoire of RL algorithms to find the optimal control policy. A broad class of RL algorithms encompassing model-based and model-free methods include:

1) **optimal control methods** – analytical methods such as linear quadratic regulator (LQR) and its variant with Gaussian noise (LQG), whereas numerical methods comprise of direct collocation, single/multiple shooting, iterative LQR [Borrelli 2011]; 2) **policy-gradient methods** – the policy parameters are iteratively improved by gradient descent on the estimated value function under the current policy [Peters 2008, Deisenroth 2013] (Least-Squares Policy Iteration (LSPI), REINFORCE, natural policy gradient, Probabilistic Inference for Learning Control (PILCO), Trusted Region Policy Optimization (TRPO), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), Relative Entropy Policy Search (REPS)); 3) **value-function methods** – the value function is approximated using methods such as dynamic programming, value-iteration, temporal difference (TD) learning and the control policy greedily follows the estimated value function [Szepesvári 2010] (Q-learning, State-Action-Reward-State-Action (SARSA), Least Squares Temporal Differences (LSTD)-λ, LSTDQ-λ); 4) **expectation-maximization methods** – the optimal policy is derived by expectation-maximization (EM) and the optimization is formulated with immediate rewards [Dayan 1997], as reward-weighted regression problem [Peters 2007], or as Policy Improvement by Path Integrals PI^2 [Theodorou 2010]; 5) **actor-critic methods** – the optimal solution is obtained by combining the policy gradient methods with the value function approximation methods to yield efficient learning with some performance guarantees [Konda 2003, Bhatnagar 2009] (actor-critic, natural actor-critic), and 6) **deep RL methods** – deep architectures are employed for computing the optimal policy [Mnih 2013, Lillicrap 2015, Duan 2016] (deep Q-Network (DQN), double Q-network, deep policy gradient (DPG), deep actor-critics).

In this section, we present several examples of computing optimal policy in continuous domains based on the task under consideration. We first present our work on actor-critics with experience replay for model-free RL in continuous challenging environments [Wawrzynski 2013], and show how the human demonstrations can be used as an initial policy for speeding up the policy search in continuous environments [Tanwani 2014].

2.3.1 Actor-Critics with Experience Replay

2.3.1.1 Classic Actor-Critic

Actor-critics are an important class of RL methods that can deal with continuous state-action space in a natural way [Kimura 1998, Konda 2003, Bhatnagar 2009]. They employ two systems, an actor and a critic. The actor represents a stochastic control policy $\boldsymbol{u}_t \sim \pi(\boldsymbol{\xi}_t; \theta)$ with parameter $\theta \in \mathbb{R}^{n_\theta}$ to generate control actions, while the critic represents the value-function approximator $\bar{V}(\boldsymbol{\xi}_t; v)$ parameterized by vector $v \in \mathbb{R}^{n_v}$. The common

method used by critic for prediction is TD learning on the value function. TD learning, or simply TD(λ), uses a weighted sum of predicted sequence of states values to estimate the *return* or sum of rewards in a given state $R^{\lambda}_{\boldsymbol{\xi}_t}$,

$$R^{\lambda}_{\boldsymbol{\xi}_t} = r_{\boldsymbol{\xi}_t} + \gamma \bar{V}(\boldsymbol{\xi}_{t+1}; v) + \sum_{i \geq 1} (\gamma\lambda)^i (r_{\boldsymbol{\xi}_{t+i}} + \gamma \bar{V}(\boldsymbol{\xi}_{t+i+1}; v) - \bar{V}(\boldsymbol{\xi}_{t+i}; v)). \tag{2.15}$$

The parameter $\lambda \in [0, 1]$ defines the dependency of the long-term reward on the predicted value-function $\bar{V}$ and the actual reward $r_{\boldsymbol{\xi}_t}$. For $\lambda = 1$, the estimator is based on true rewards and thus unbiased, but its variance may be very high, while $\lambda = 0$ ensures low variance at the cost of high bias of the value-function approximator. TD(0) is also commonly known as value iteration and TD(1) is the Monte-Carlo estimation.

The algorithm works in combination such that the actor generates the controls stochastically and the critic predicts the expected value of $R^{\lambda}_{\boldsymbol{\xi}_t}$. Let $\phi(\boldsymbol{\xi}_t, \theta, v)$ and $\psi(\boldsymbol{\xi}_t, \theta, v)$ define the average direction of improvement along the vectors parameterizing the actor and the critic respectively. A visit in state $\boldsymbol{\xi}_t$ modifies the policy vector θ along the estimator $\hat{\phi}(\boldsymbol{\xi}_t, \theta, v)$ that defines the gradient of the true expected return $R^{\lambda}_{\boldsymbol{\xi}_t}$:

$$\begin{aligned} \hat{\phi}(\boldsymbol{\xi}_t, \theta, v) &= (R^{\lambda}_{\boldsymbol{\xi}_t} - \bar{V}(\boldsymbol{\xi}_t; v)) \nabla_\theta \ln \pi(\boldsymbol{\xi}_t; \theta) \\ \theta &= \theta + \beta^{\theta}_t \, \hat{\phi}_t(\boldsymbol{\xi}_t, \theta, v) \end{aligned} \tag{2.16}$$

where step-size β^{θ}_t is a small positive number, and ∇_θ is the gradient with respect to θ. If the action yields the return $R^{\lambda}_{\boldsymbol{\xi}_t}$ larger than the approximated value $\bar{V}(\boldsymbol{\xi}_t; v)$, the probability of selecting action $\boldsymbol{u}_t$ in state $\boldsymbol{\xi}_t$ is increased. If, conversely, the action turns out to bring rewards smaller than expected, then its probability is being decreased. The critic vector is accordingly adjusted to minimize the discrepancy between $R^{\lambda}_{\boldsymbol{\xi}_t}$ and $\bar{V}(\boldsymbol{\xi}_t; v)$, given by its gradient estimate $\hat{\psi}(\boldsymbol{\xi}_t, \theta, v)$. The update is given by the product of estimate $\hat{\psi}(\boldsymbol{\xi}_t, \theta, v)$ and small positive step-size β^{v}_t:

$$\begin{aligned} \hat{\psi}(\boldsymbol{\xi}_t, \theta, v) &= (R^{\lambda}_{\boldsymbol{\xi}_t} - \bar{V}(\boldsymbol{\xi}_t; v)) \nabla_v \bar{V}(\boldsymbol{\xi}_t; v) \\ v &= v + \beta^{v}_t \, \hat{\psi}(\boldsymbol{\xi}_t, \theta, v) \end{aligned} \tag{2.17}$$

2.3.1.2 Actor-Critics with Experience Replay

The idea of *experience replay* [Cichosz 1999, Wawrzynski 2013, Lillicrap 2015, Malla 2017] is to use previously collected samples to intensify the learning process of the original sequential algorithm as if the events have just happened. In the classic actor-critic algorithm described above, the policy vector is adjusted after every time instant t along the estimate

of policy improvement $\phi(\boldsymbol{\xi}_t, \theta, v)$. Whereas in the actor-critic algorithm with experience replay, the actor repeatedly chooses one of the recently visited states policy improvement estimate $\phi(\boldsymbol{\xi}_i, \theta, v)$ within each time instant t, and modifies the current policy vector along the estimate. The actor employs the gathered experience to adjust different policies characterized by different policy vectors. The critic undergoes the same operation as that of the actor in the modified algorithm with experience replay. Essentially both algorithms achieve the same goal, but the modified one improves the current actor and critic with the use of the whole gathered experience rather than only the present event as in the classic method. Due to more exhaustive exploitation of information, experience replay leads to faster learning at the cost of additional computation.

The concept of reusing samples evolved from the *importance sampling* technique [Sutton 1998]. Although the bias vanishes asymptotically during re-sampling of the previous states, the variance of the actor-critic estimators significantly increases, thereby, limiting its use for RL control tasks. In [Wawrzyński 2009], adaptive importance sampling with *randomized-truncated estimators* is used to reduce the variation of the estimators while re-sampling the previously visited states. This ensures stability of the process while allowing the past experience to intensify the learning process. The randomized truncated estimators of $\hat{\phi}_r(\boldsymbol{\xi}_i, \theta, v)$ and $\hat{\psi}_r(\boldsymbol{\xi}_i, \theta, v)$ appropriately compensate for the fact that the current policy is different from the one that generated the actions in the database. They are given by,

$$\hat{\phi}_r(\boldsymbol{\xi}_i, \theta, v) = \nabla_\theta \ln \pi(\boldsymbol{\xi}_t; \theta) \sum_{k=0}^{K} \alpha_r^k d(\boldsymbol{\xi}_{i+k}; v) \min \left\{ \prod_{j=0}^{k} \frac{\pi(\boldsymbol{\xi}_{i+j}, \theta)}{\pi(\boldsymbol{\xi}_{i+j}, \theta_{i+j})}, b \right\}, \tag{2.18}$$

$$\hat{\psi}_r(\boldsymbol{\xi}_i, \theta, v) = \nabla_v \bar{V}(\boldsymbol{\xi}_t; v) \sum_{k=0}^{K} \alpha_r^k d(\boldsymbol{\xi}_{i+k}; v) \min \left\{ \prod_{j=0}^{k} \frac{\pi(\boldsymbol{\xi}_{i+j}, \theta)}{\pi(\boldsymbol{\xi}_{i+j}, \theta_{i+j})}, b \right\}, \tag{2.19}$$

where $b > 1$, $\alpha_r \in [0, 1)$, θ_{i+j} is the policy vector to generate $\boldsymbol{u}_{i+j}$, K is a positive integer random variable drawn independently from a geometric distribution $Geom(\rho)$[3] with parameter $\rho \in [0, 1)$, and $d(\boldsymbol{\xi}_i; v)$ is the TD(0) of the form

$$d(\boldsymbol{\xi}_i; v) = r_{\boldsymbol{\xi}_i} + \gamma \bar{V}(\boldsymbol{\xi}_{i+1}; v) - \bar{V}(\boldsymbol{\xi}_i; v). \tag{2.20}$$

For implementation, we split the algorithm into two threads executing simultaneously: *control thread* - one that controls the robot by sampling actions, and *actor critics optimization thread* - one that optimizes the parameters of the actor-critic networks. The step sizes of the actor-critic networks, β_t^θ and β_t^v, are updated online by fixed point method of step-size

[3] K has a geometric distribution $Geom(\rho)$ with $P(K = m) = (1 - \rho)\rho^m$ for positive integer m.

Algorithm 2 *Actor-Critics with Experience Replay*

Input: Initial actor policy vector θ_0, critic vector v_0, computation steps n_c

procedure `CONTROL`

$N_s = 0$
repeat
  Draw and execute action, $\boldsymbol{u}_t \sim \pi(\boldsymbol{\xi}_t; \theta)$
  Register the sample $< \boldsymbol{\xi}_t, \boldsymbol{u}_t, \theta, r_{\boldsymbol{\xi}_t}, \boldsymbol{\xi}_{t+1} >$ in the database
  Increase the computation steps, $N_T = N_T + n_c$
until the optimal policy is found $\pi \approx \pi^*$

procedure `ACTOR CRITICS OPTIMIZATION`

$k := 0$
loop
  while there are pending total computation steps ($k \leq N_T$) **do**
    Make sure only N most recent samples are present in the database
    Draw $i \in \{t - N + 1, t - N + 2, \ldots, t\}$
    Adjust θ along an estimator of $\phi(\boldsymbol{\xi}_i, \theta, v)$ (see Eq. (2.18))
    $\theta := \theta + \beta_t^\theta \hat{\phi}_r(\boldsymbol{\xi}_i, \theta, v)$
    Adjust v along an estimator of $\psi(\boldsymbol{\xi}_i, \theta, v)$ (see Eq. (2.19))
    $v := v + \beta_t^v \hat{\psi}_r(\boldsymbol{\xi}_i, \theta, v)$
    $k := k + 1$
  end while
end loop

estimation [Wawrzynski 2013]. The overall actor-critic algorithm with experience replay is presented in Algorithm 2.

2.3.2 Octopus Arm and Half-Cheetah Examples

We now demonstrate the effectiveness of our algorithm on simulated control problems in continuous domains. We choose two diverse and challenging tasks to this end: 1) point reaching movement of octopus arm, and 2) cyclic running motion of half-cheetah.

2.3.2.1 Octopus Arm Example

Octopus is well-known for exhibiting a high level of flexibility in controlling arm movements [Yekutieli 2005]. The highly developed limbs of octopus make it capable of bending, stretching, shortening and twisting its arm at any point and in any direction with virtually unlimited degrees of freedom. A cross-sectional examination of an octopus arm reveals that the muscles alone — without any rigid skeleton — are responsible for providing the

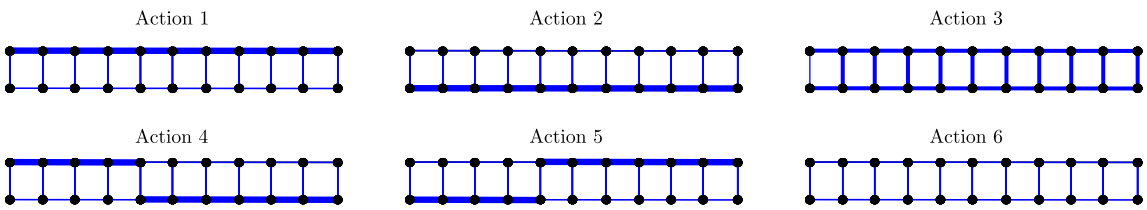


Figure 2.6: Octopus arm action set: the activated muscles are indicated by thick blue lines.

structural support and generating the movement of arm. This allows the octopus arm to execute dexterous motor control tasks that are unprecedented for other biological and artificial systems. We are interested in learning a reactive control policy for the octopus arm to reach an arbitrary goal point starting from some random position.

State-Action Space and Reward Function: The octopus arm is represented by a planar simulator model composed of 10 quadrilateral compartments with fixed base, each muscle 1-unit long with action duration 0.1 seconds, as described in [Yekutieli 2005]. We associate 3 frames of reference in polar coordinates that are used to define features to localize the octopus arm movement towards its goal. The first frame is located at the center of the point masses of all the quadrilateral compartments, the second one is located at the center of the point masses of last 5 quadrilateral compartments, and the third one is located at the center of mass of the last compartment. Based on the three frames and the goal frame, the state space of our model consists of 12 state variables, normalized to roughly cover the interval $[-1, 1]$. The action space consists of a set of 6 actions each of which pre-defines the activation level in the arm's muscles, as used in [Engel 2005]. The action space is shown in Figure 2.6.

The learning task is carried out in episodes. An episode terminates with success when the last compartment touches the goal. If this goal is not reached within 500 steps, the episode terminates. The reward function is defined such that the controller is penalized with a negative score of -1 for all the learning steps in which the goal has not been reached. Moreover, the arm is rewarded for moving towards the goal in proportion to the velocity of the last compartment in the direction of the goal. The goal of the octopus arm is to maximize the reward function by reaching the goal position as quickly as possible.

Actor and Critic Structure: The actor and critic are based on feedforward neural networks, namely 2-layer perceptrons with sigmoidal neurons in their hidden layers and linear neurons in their output layers. The initial weights in the hidden layers are drawn randomly from the normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and the weights of the output layers are initially set to zero.

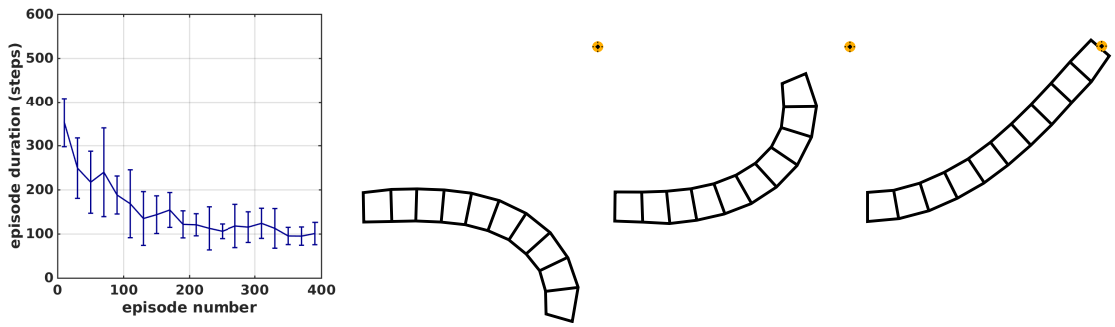


Figure 2.7: Octopus arm results. *(left)* episode number against duration steps of each episode, *(right)* simulated goal reaching movement of octopus arm at the start of episode; with arm position trying to reach the goal; with arm position at the goal. After learning, the arm reaches the goal in about 100 steps.

The actor is the combination of a neural network and a generator of discrete numbers. The network has N_A neurons in its hidden layer and 6 neurons in the output layer corresponding to the size of the discrete action set. The critic is a 2-layer perceptron with N_C neurons in its hidden layer and one neuron in the output layer.

Results and Discussions: The parameter setting of the actor-critic reinforcement learning algorithm is as follows: $N_A = 50$, $N_C = 100$, $\gamma = 0.98$, $b = 2$, $\alpha_r = \gamma = 0.98$, $n_c = 100$ and $\lambda = 0.5$. Figure 2.7 shows different stages of the octopus arm in reaching the desired goal. On the left side, the figure presents the learning curve (average rewards vs. episode number). It can be seen that the learning converges after 240 episodes which is equivalent to 80 minutes of Octopus time.

2.3.2.2 Half-Cheetah

Half-Cheetah is a 6 degrees of freedom planar robot, introduced in [Wawrzyński 2009]. It is composed of nine links, eight joints and two paws (see Figure 2.8). The angles of the fourth and fifth joint are fixed while others are controllable. The torque applied at each joint acts as input to the model and the next position of the robot is obtained as output by integrating its dynamic equations of motion. The control problem is to learn a reactive policy under the MDP framework to make half-cheetah run as fast as possible.

State-Action Space and Reward Function: The state of half-cheetah is defined by 31 variables. The action space is continuous, contrary to that of octopus arm, with 6 dimensions each corresponding to one actuated joint independently. Learning is divided into episodes with an average duration of 250 steps, for 0.02 second duration of each step.

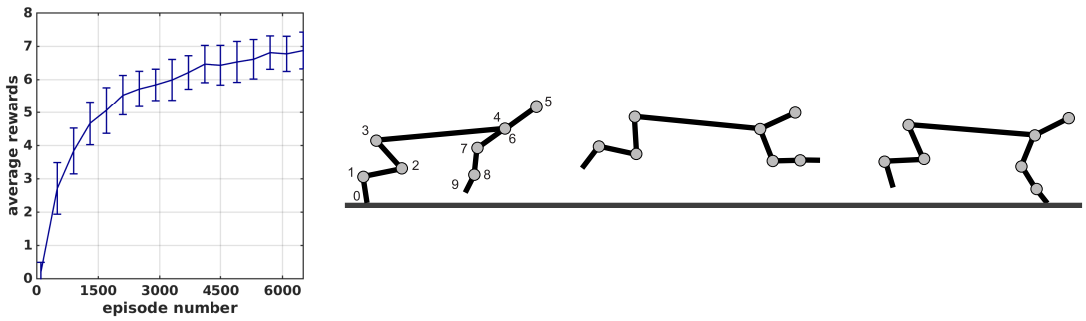


Figure 2.8: Half-Cheetah results. *(left)* episode number against the average rewards in each episode; *(right)* simulated run of Half-Cheetah with initial stance, middle stance with feet in the air, and landing stance. After learning, half-cheetah runs with a speed of about 6.5 m/sec.

The robot is mainly rewarded for its speed of moving forward. Other components are minor penalties for violating torque limits, joint limits, not moving the trunk in idle position and touching the ground with other body parts than paws.

Actor and Critic Structure: The actor is composed of two parts: a neural network and a normal distribution. The input of the network is the state of half-cheetah. The network has a hidden layer with N_A sigmoidal neurons and six linear output neurons corresponding to the dimensionality of the action space. The output of the network becomes a mean value of the normal distribution with unit covariance matrix to generate exploratory control actions. The critic is a 2-layered neural network with N_C neurons in its hidden layer and 1 neuron in the output layer.

Results and Discussions: The experiments to make half-cheetah run are configured with the following parameters: $N_A = 160$, $N_C = 160$, $\sigma = 5$, $\gamma = 0.99$, $b = 2$, $\alpha_r = \gamma = 0.99$, $\rho = \lambda = 0.9, n_c = 30$. Figure 2.8 shows various stages of the learned running gait of half-cheetah. The cat model starts from a standing position and first learns to move forward with the added noise in the control system. The awkward walk transforms into a trot gait which at the end of training becomes a smooth nimble run. The use of experience replay speeds up the learning process of running in proportion to the intensity of replaying computations. The figure also shows that the algorithm requires 3000 episodes (about 4.2 hours of half-cheetah time) to learn to receive average reward of 6, which corresponds to a nimble run.

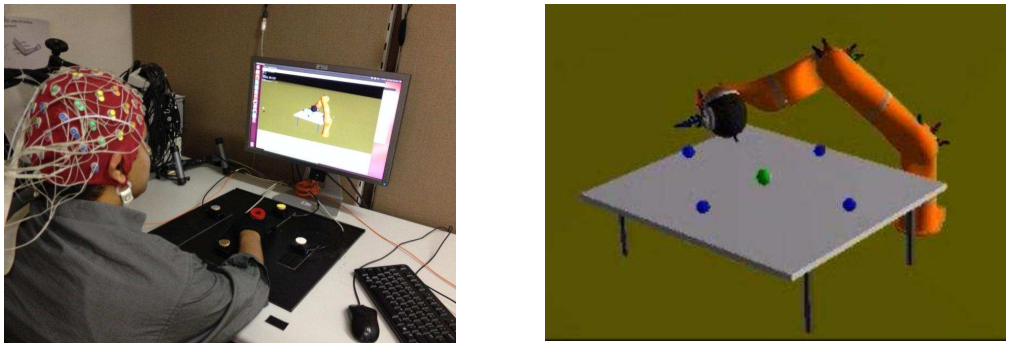


Figure 2.9: Decoding goal from slow cortical EEG signals on left which moves the KUKA robot arm optimally to the desired target on right.

2.3.3 Learning with Initial Policy

Considering the amount of time it takes to find an optimal policy in continuous domains, a lot of effort is made in the RL and robotics community to incorporate prior knowledge about the task and speed up the learning process. Most of the work has focused on using the demonstrations provided by the human/simulator to initialize the control policy of the task, also called the primitive policy or the fixed policy. Human demonstrations can also be used for initializing the value function and/or learning the transition dynamics model. The initial policy provides an educated initial guess for the leaner to confine its search to near optimal regions. The use of initial policy makes the optimization problem feasible in higher dimensional spaces [Kawato 1999, fang Wang 2016].

Applications include online trajectory modulation with obstacles [Guenter 2007], incorporating dynamic movement primitives for Ball-in-a-Cup task [Kober 2010] and pancake flipping task [Kormushev 2010], bimanual rod manipulation to hit via-points [Sugimoto 2013], and optimizing walking gait of a humanoid robot [Tanwani 2011]. Below we present an application of decoding the EEG signals of the human to reach a desired goal, where the initial policy learned from human demonstrations is combined with RL to yield better control policies [Tanwani 2014].

2.3.4 Decoding EEG Signals for Arm Control Example

Decoding the user intention from non-invasive EEG signals is a challenging problem. We study the feasibility of predicting the goal for rewards-driven control of the robot arm in self-paced reaching movements, i.e., spontaneous movements that do not require an external cue. Contrary to decoding the cue-based movements [Musallam 2004, Waldert 2008], we

consider self-paced reaching movements where the user spontaneously executes the movement without an external cue [Niazi 2011]. Intention here refers to the high-level target of the user as compared to the low-level muscle activations for executing the movement. The integrated framework combines the high-level goals encoded in EEG signals with low-level motion plans to control the robot arm in continuous task space. A promising application of this work is to use EEG signals for direct motor control of patients with possibly upper-limb disabilities. The overall system is presented in Fig. 2.9. EEG signals of the user give the intended goal of moving to one of the four targets in cardinal directions by online classification. The desired goal fed to the optimal motion plans generator to move the arm towards the desired target.

Dataset: The dataset used here was designed to perform center-out planar reaching movements to four goal targets in cardinal directions located 10 cm away from the center, while holding the PHANTOM robotic arm. Four subjects – two healthy and two stroke patients – participated in the experiment carried out at the San Camillo Hospital, Venice, Italy. One patient had left paretic arm with left cerebellar hemorrhagic stroke since 2 months; while other had right paretic arm suffering from left nucleo-capsular stroke since 2 years. After the target was shown to the subject, the subject was asked to wait for at least 2 seconds to perform a self-paced movement (see [Lew 2012] for details of experimental set-up). For each arm, subjects performed three runs each containing 80 trials each (20 trials per target). Trials were extracted ranging from 2 s before the movement onset until 1 s after the task. For brevity, we only report results of the right arm of the first healthy subject in this work.

The EEG and EOG signals were simultaneously recorded with a portable BioSemi ActiveTwo system using 64 electrodes arranged in an extended 10/20 montage. EOG channels were placed above nasion and below the outer canthi of both eyes in order to capture horizontal and vertical EOG components. The kinematics data of the robotic arm was recorded at 100 Hz, while EEG signals were captured at 2048 Hz and then downsampled to 256 Hz. Preprocessing steps to analyse EEG data required Common Average Referencing (CAR) procedure to remove the global background activity [Bertrand 1985]. Moreover, only 34 EEG channels were selected, excluding the peripheral channels and those having high correlation with the EOG activity. EEG signals were then passed through a zero-phase low-pass Butterworth filter with cut-off frequency of 120 Hz, further down-sampled at 128 Hz and finally low-pass filtered at 1 Hz to extract slow cortical potentials. Each EEG channel and kinematic signal was normalized to have zero-mean and unit-standard deviation.

2.3.4.1 Intention Decoding

We perform online classification in a sliding window of 250 ms that shifts by 62.5 ms within the trial period of $[-2\ 1]$ seconds to decode the intention/goal . Note that we start to decode the goal prior to the movement onset to minimize any delays in controlling the arm (see [Lew 2012] for details). For each of these windows, the features are selected separately using Canonical Variant Analysis (CVA) with 5 fold cross-validation taking one EEG sample per window at the end. 10 EEG channels with best discriminant power are selected in each window to classify among the 4 target goals. For classification, EEG data is further downsampled to 16 Hz taking into account 4 samples of 10 EEG channels for a total of 40 features. Linear Discriminant Analysis (LDA) [Duda 2000] is then used for predicting the i-th goal estimate $\boldsymbol{\xi}_g$ in every time window from the given EEG feature vector. For the EEG feature vector represented by $\boldsymbol{u}_t$ at time instant t, the classification of the goal $\boldsymbol{\xi}_{g_t}$ is based on the probability of belonging to each of the goals

$$\boldsymbol{\xi}_{g_t} = g(\boldsymbol{u}_t) = \arg\max_{i=1\ldots4} \mathcal{P}(\boldsymbol{\xi}_g^{(i)}|\boldsymbol{u}_t). \tag{2.21}$$

2.3.4.2 Trajectory Decoding

We want to continuously generate the motion plans to drive the robot arm to the goal in an optimal manner. We represent this decoder with a dynamical system of the form,

$$\dot{\bar{\boldsymbol{\xi}}}_t = f(\bar{\boldsymbol{\xi}}_t; \theta) + \varepsilon, \tag{2.22}$$

where f is a continuously differentiable function that maps the 2D-planar Cartesian position of the robot arm $\boldsymbol{\xi}_t$ to its Cartesian velocity $\dot{\boldsymbol{\xi}}_t$, and $\theta \in \mathbb{R}^{n_\theta}$ represent the parameters of the function f. The function f here represents the control policy of the robot and maps the current position of the robot to the velocity which in turn gives the next desired position upon integration. For ease of computation, we transform the coordinates to $\bar{\boldsymbol{\xi}}_t = \boldsymbol{\xi}_t - \boldsymbol{\xi}_g$ to signify the change of all goal positions to the origin of the transformed system (see Fig. 2.12 for the transformed demonstrations pointing to the origin). We are required to learn the parameters θ such that the robot follows the intended movement of the user. Note that the encoding of the demonstrations using any function approximator typically creates spurious attractors or divergent behaviour away from the training data [Khansari-Zadeh 2010]. Often stability conditions are required [Khansari-Zadeh 2011] and even if the resulting dynamical system is stable, it may not get the user to the desired goal in finite amount of time. To this end, we take a two-step methodology: 1) learn the initial function from demonstrations of the hand kinematics recorded from the subject, and 2)

optimize the function parameters by RL for effective generalization and generating optimal motion plans [Sutton 1998].

Initial Dynamical System: In the first stage, we use Support Vector Regression (SVR) to estimate the initial function in Eq. (2.22) given data samples $\{\bar{\boldsymbol{\xi}}_t, \dot{\bar{\boldsymbol{\xi}}}_t\}$ from the experiments. Note that each output dimension is learned separately in this model. To speed up the learning process, we downsample the kinematic data to 5 Hz for a total of 750 samples corresponding to the right arm of the first subject in the training set. Hyper-parameters of the SVR are obtained after grid-search with size of the epsilon-tube, $\varepsilon = 0.5$, width of the radial basis kernel function $\gamma = 0.5$, and complexity parameter $C = 1$.

Optimized Dynamical System: In the second stage, we modify the landscape of the learned function to learn optimal policy in the whole state space by maximizing the reward function. This would enable the robot to decode the movement effectively far from the training data (see Fig. 2.12 for clarity). Moreover, optimization in the second stage caters for the imperfection or sub-optimality in the recorded demonstrations (for example, demonstrations of stroke suffering subjects). We express the reward function $r_{\bar{\boldsymbol{\xi}}_t}$ as

$$r_{\bar{\boldsymbol{\xi}}_t} = - w_1 \bar{\boldsymbol{\xi}}_T^{\top} \bar{\boldsymbol{\xi}}_T \; - w_2 \dot{\bar{\boldsymbol{\xi}}}_T^{\top} \dot{\bar{\boldsymbol{\xi}}}_T - w_3 \ddot{\bar{\boldsymbol{\xi}}}_t^{\top} \ddot{\bar{\boldsymbol{\xi}}}_t, \tag{2.23}$$

where w_1 weighs the cost for distance from the goal/origin at the end of the trial, w_2 penalizes for any non-zero velocity at the end of the trial, and w_3 is responsible for ensuring smooth movement in reaching the goal by minimizing the norm of the acceleration vector. Weights of the reward function after manual tuning are: $w_1 = 5,\ w_2 = 0.01,\ w_3 = 0.0001$. Maximum velocity $\dot{\boldsymbol{\xi}}_{max}$ is set to 30 cm/s^2 and the simulations are carried till $T = 2$ seconds to prolong the penalty by w_1 and w_2 after the end of trial at $t = 1$ second.

Support vectors of the initial function act as basis functions to optimize the function f in the second stage. Weights of the support vectors θ are optimized by stochastic gradient ascent on the value function, $V(\bar{\boldsymbol{\xi}}_t; \theta) = \frac{1}{T}\sum_{t=0}^{T} r_{\bar{\boldsymbol{\xi}}_t}$ starting from the initial state $\boldsymbol{\xi}_0$ and integrating the dynamics model $\dot{\bar{\boldsymbol{\xi}}}_t = f(\bar{\boldsymbol{\xi}}_t; \theta)$. Note that we do not use a function approximator to represent the value function, and take the gradient of the estimated value function as in policy gradient approaches. More precisely, we add noise η sampled from multivariate normal Gaussian $\mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$ with $\sigma = 0.1$ to the parameters θ, evaluate the value function, $V(\bar{\boldsymbol{\xi}}_t; \theta + \eta)$, from episodic roll-outs of the current optimized function, $\dot{\bar{\boldsymbol{\xi}}}_t = f(\bar{\boldsymbol{\xi}}_t; \theta + \eta)$, and adjust the parameter vector in the direction of increasing value function, i.e.,

$$\theta = \theta + \beta_t^{\theta} \left(V(\bar{\boldsymbol{\xi}}_t; \theta + \eta) - V(\bar{\boldsymbol{\xi}}_t; \theta) \right), \tag{2.24}$$

where β_t^{θ} is a small step-size parameter set to 0.05 in our experiments. The procedure

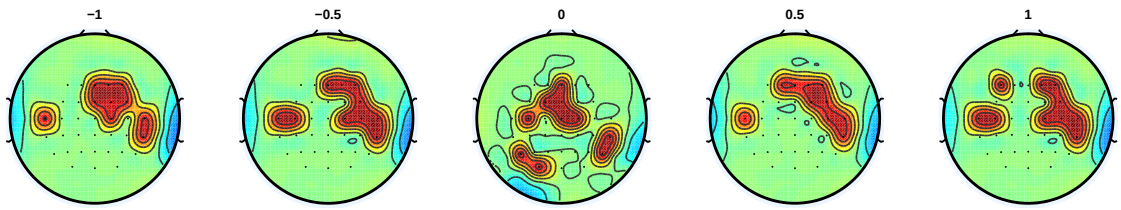


Figure 2.10: Evolving EEG channels activity in the time interval $[-1\ 1]$ seconds.

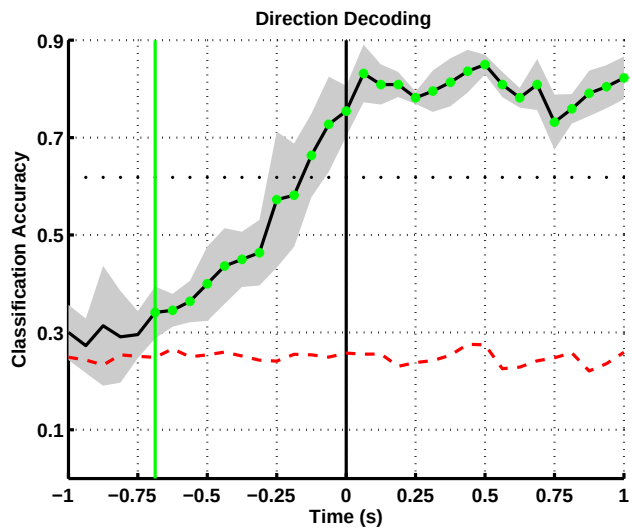


Figure 2.11: Decoding goal direction from EEG signals of first healthy subject (right arm). Red line shows the chance level; green line indicates the time instant when the classification accuracy significantly exceeds the chance level; shaded region shows the variation in accuracy over 5-folds.

is repeated till the parameter vector stops changing. In our experiments, the parameter vector is improved for 1500 iterations which increases the value of the function parameters $V(\theta)$ from -118.1 to -4.81. In the proposed framework, the attractor of the optimized dynamical system is shifted from the origin to the estimated goal from Eq. (2.21) which is updated after every time window of 250 milliseconds. After the end of trial, the optimized dynamical system moves the robot arm to the last estimated goal at $t = 1$ seconds. The optimized dynamical system/policy takes the form,

$$\dot{\boldsymbol{\xi}}_t = f(\bar{\boldsymbol{\xi}}_t + \boldsymbol{\xi}_{g_t}). \tag{2.25}$$

Results and Discussions: To analyse the performance of the goal decoding from EEG signals, we show the topographic plots of selected channels to depict their discriminatory power at different time instants starting 1 second before the movement onset in Fig. 2.10.

As the exact time when movement intent occurs in a self-paced movement is unclear, the plots provide insights about movement-related modulations in different brain regions during planning and how they evolve over time. It is seen that the activity is dominant in the frontal-parietal regions of brain consistent with earlier reported studies [Lew 2012].

Fig. 2.11 reports the classification accuracy of goal decoder in the time window $[-1\ 1]$ seconds. Classification accuracy is computed as the ratio between the sum of correctly classified diagonal entries in the confusion matrix and the total number of instances. The time instant when the classification accuracy significantly exceeds the chance level is used as a metric to initiate the movement with the trajectory decoder. Chance level is calculated by training the classifier on a randomized permutation of the class labels of the training set. Results are then averaged across 10 iterations each with 5 fold cross-validation. Best time for subject 1 is 687.5 ms with classification accuracy of 0.34 before the movement

Table 2.1: Performance comparison of initial and optimized dynamical system using: MSE on the testing set; average correlation in time between simulated and demonstrated position trajectories on the testing set; end-point distance from the goal for different initial conditions

Trajectory Decoder	**MSE** cm/s^2	**Correlation** $[0\ 1]$	**End-Point Distance** (cm)
Initial SVR	2.49	0.51	5.157
Optimized SVR	-	0.23	0.09

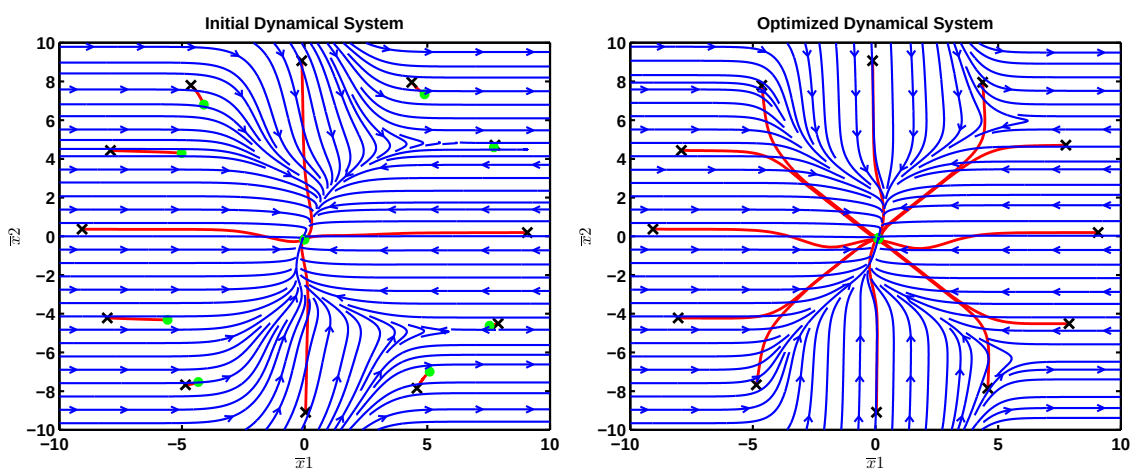


Figure 2.12: Evaluation of the policy from different initial conditions: *(left)* initial learned dynamical system function with SVR in a supervised manner, *(right)* optimized dynamical system with stochastic gradient ascent. Black crosses indicate the initial positions, while green circles denote the position at the end of the trial

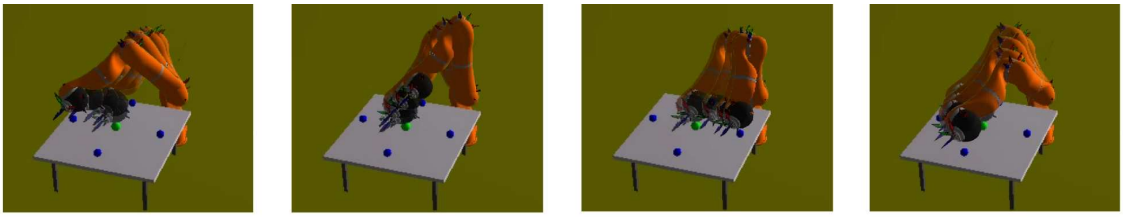


Figure 2.13: Simulated trajectories of KUKA robot performing center-out reaching task

onset (marked with green line in Fig. 2.11). It is seen that the classification accuracy gradually improves afterwards with a peak accuracy of 0.85 at 0.5 seconds. We evaluate the performance of our trajectory decoder using three metrics: 1) Mean-Square Error (MSE) on the training/testing set, 2) Correlation in time of the simulated position trajectories with the demonstrated ones, 3) Distance to the goal at the end of the trial computed by simulating the system from 12 different initial conditions. Table 2.1 summarizes the performance of the initial and the optimized SVR. The initial dynamical system learned using SVR performs well in terms of MSE with training and testing error of 2.66 and 2.49 cm/s^2 respectively, and a high correlation in position of 0.51 with the demonstrated trajectories. To evaluate the performance of the system far from the training data, we sample 12 different initial points in the plane (shown in Fig. 2.12 with crosses) and integrate the system forward in time for a period of 2 seconds. As seen in Fig. 2.12, the initial dynamical system with SVR is not able to generalize away from the training data yielding a high end-point distance error of 5.157 cm. Note that the initial conditions in the cardinal directions correspond to the training set. On the other hand, optimized SVR is able to drive the robot arm to the goal from all the sampled initial conditions. This comes at a cost of relatively low position correlation of 0.23 suggesting the need to further improve the reward function. This generalization is required in our application since the user is expected to control the arm from all parts of the state space.

The desired goal is inferred from the EEG signals and the trajectory decoder optimally guides the robot arm to the desired goal. We test the performance of the integrated system on the simulated 7 degrees of freedom KUKA robotic arm as shown in Fig. 2.14. The optimized dynamical system starts to move the robot arm 687.5 milliseconds before the movement onset on average and guides the robot arm to the last estimated goal at the end of the trial. Across all the trials, the robot arm reaches the actual goal with a net accuracy of 79.5% on average. The figure shows simulated trajectories of the robotic arm reaching different goal positions following the predicted goal from the intention decoder and the optimal motion plans from the trajectory decoder.

In the aforementioned work, the intention recognition is posed as an online classification

problem. We now investigate the IRL problem in continuous unknown environments where the intention is described in the form of a reward function that the user optimizes for in the demonstrations.

2.4 Inverse Reinforcement Learning in Continuous Domains

Performing IRL in continuous domains is computationally more demanding because of the time it takes to find an optimal policy in continuous high-dimensional state-action spaces and the liability of most IRL algorithms to repeatedly find an optimal policy for extracting the reward function. In [Aghasadeghi 2011], the authors follow the maximum entropy formulation (see Eq. (2.6)) and iteratively improve the approximation of the partition function in continuous state-space with a set of optimally sampled trajectories for the current estimate of the reward function. Kalakrishnan *et al.* used l_1 norm regularization on the reward function parameters to discard the effect of the redundant features with path integral IRL [Kalakrishnan 2013]. Boularias *et al.* extended the maximum entropy framework to estimate the reward function parameters in a model-free setting by minimizing the KL divergence between the demonstrated and the derived policy [Boularias 2011]. Levine and Koltun apply the Laplace approximation to the partition function in the maximum entropy IRL corresponding to locally solving the optimal control problem [Levine 2012]. The approximation allows to optimize the reward function parameters directly and prevent the need of repeatedly finding an optimal policy in the inner loop of a candidate reward function. In our work presented in [Tanwani 2013a], we demonstrate the use of IRL in continuous unknown environments for a special class of trajectory reward functions that penalizes any deviation from a desired reference trajectory.

2.4.1 Model-Based Learning for Trajectory Reward Function

IRL is desirable for estimating the human preferences in the demonstrations for various task-dependent objectives, and learning rich control policies that generalize beyond the demonstrated data. We divide our framework in three stages for learning the main constituent components of IRL in an iterative manner: *dynamics model*, *reward function* and *optimal policy*.

2.4.1.1 Dynamics Model Learning

Given the set of all demonstrations $\{\boldsymbol{\xi}_t, \boldsymbol{u}_t\}_{t=1}^T$, we first learn the dynamics model of the form, $\dot{\boldsymbol{\xi}} = f(\boldsymbol{\xi}_t, \boldsymbol{u}_t; \theta)$, parameterized by vector $\theta \in \mathbb{R}^{n_\theta}$ in a supervised learning manner. An interested reader can find a review of function approximation methods for supervised learning in [Stulp 2015].

2.4.1.2 Reward Function Learning

For many manipulation tasks, it is difficult to specify high-level objectives that humans optimize for during the task. We represent the reward function for such tasks with a desired reference trajectory (known or unknown) that the humans follow in their demonstrations, and penalize any deviation from this reference trajectory in a quadratic manner, namely

$$r(\boldsymbol{\xi}_t, \boldsymbol{u}_t) = -(\boldsymbol{\xi}_t - \boldsymbol{\xi}_t^*)^\top \boldsymbol{Q}(\boldsymbol{\xi}_t - \boldsymbol{\xi}_t^*) - (\boldsymbol{u}_t - \boldsymbol{u}_t^*)^\top \boldsymbol{R}(\boldsymbol{u}_t - \boldsymbol{u}_t^*), \tag{2.26}$$

where $\boldsymbol{\xi}_t^*$ and $\boldsymbol{u}_t^*$ are the reference trajectories, $\boldsymbol{Q} \succeq 0$ and $\boldsymbol{R} \succ 0$ are diagonal matrices containing the reward function parameters $\boldsymbol{w}$ with $\boldsymbol{Q} = \text{diag}\begin{bmatrix} w_1 & \dots & w_D \end{bmatrix}, \boldsymbol{R} = \text{diag}\begin{bmatrix} w_{D+1} & \dots & w_{D+m} \end{bmatrix}$. Note that this reward function is maximized when the optimal control policy and the desired reference trajectory match with each other. Atkenson and Schaal in [Atkeson 1997a] used the human demonstration as a trajectory reward function with hand tuned gains and learned the dynamics model incrementally with the data collected from successive attempts to perform the task of pendulum swing-up. The authors in [Coates 2008] recover a single trajectory from multiple demonstrations that is consistent with the task dynamics. The reward function and the current task model is used to generate the control policy via trajectory optimization.

Here, we follow the approach in [Levine 2012] to recover the reward weights $\boldsymbol{w}$ under which the observed human demonstrations $\mathcal{D}_E^\pi$ are locally optimal for the given dynamics model under the maximum entropy distribution (see Eq. (2.6)). The approach requires the human demonstrations to only be locally optimally with respect to the underlying reward function. By approximating the integral in the partition function locally around the expert demonstration using the deterministic Laplace method, the log-likelihood function of the reward function can be computed as

$$\mathcal{L}(\boldsymbol{w}|\mathcal{D}^{\pi_E}) = \frac{1}{2}\mathbf{g}^T\mathbf{H}^{-1}\mathbf{g} + \frac{1}{2}\log|-\mathbf{H}|, \tag{2.27}$$

where $\mathbf{g} = \frac{\partial V^\pi}{\partial \pi}$ and $\mathbf{H} = \frac{\partial^2 V^\pi}{\partial \pi^2}$ is evaluated around the locally optimal human demon-

stration with $\pi = \{\boldsymbol{u}_1^E, \boldsymbol{u}_2^E, \ldots, \boldsymbol{u}_T^E\}$. Computing this likelihood requires the gradient and the Hessian of the value function, which in turn requires linearizing the dynamics along the expert demonstration. The likelihood function is then maximized using gradient-based optimization to solve for the reward function parameters $\boldsymbol{w}$. Note that we did not require a repeated optimal policy solver to find the locally optimal reward function. However, the inaccuracy of the dynamics model (due to insufficient number of samples) may require re-estimation of the dynamics model and subsequently, the reward function.

2.4.1.3 Optimal Policy Learning

We use the iterative linear quadratic regulator (iLQR) [Li 2004] to learn the optimal trajectory for the estimated dynamics model and the reward function. Starting from a random policy, iLQR successively improves the policy estimate by solving a LQR problem in an inner loop with quadratic approximation of the reward function and linear approximation of the dynamics model along the current optimal trajectory. The steps can be highlighted as: 1) execute the current policy $\pi^{(j)}$ starting from $j = 1$ and record the resulting state-input trajectory $\{\boldsymbol{\xi}_t, \boldsymbol{u}_t\}_{t=1}^T$, 2) evaluate the first order partial derivatives of the estimated dynamics model $\{\frac{\partial f}{\partial \boldsymbol{\xi}_t}, \frac{\partial f}{\partial \boldsymbol{u}_t}\}_{t=1}^T$ and the second order partial derivatives of the estimated reward function $\{\frac{\partial r_{\boldsymbol{\xi}_t}}{\partial \boldsymbol{\xi}_t}, \frac{\partial r_{\boldsymbol{\xi}_t}}{\partial \boldsymbol{u}_t}, \frac{\partial^2 r_{\boldsymbol{\xi}_t}}{\partial \boldsymbol{\xi}_t^2}, \frac{\partial^2 r_{\boldsymbol{\xi}_t}}{\partial \boldsymbol{u}_t^2}, \frac{\partial r_{\boldsymbol{\xi}_t}}{\partial \boldsymbol{u}_t \partial \boldsymbol{\xi}_t}\}_{t=1}^T$ along the trajectory $\{\boldsymbol{\xi}_t, \boldsymbol{u}_t\}_{t=1}^T$, 3) find the optimal policy $\pi^{(j+1)} = \hat{\boldsymbol{u}}_t + \boldsymbol{K}_t(\hat{\boldsymbol{\xi}}_t - \boldsymbol{\xi}_t)$ for linear quadratic system in step 2, where $\hat{\boldsymbol{u}}_t, \hat{\boldsymbol{\xi}}_t$ correspond to the open-loop corrected trajectories and $\boldsymbol{K}_t$ gives the variable stiffness and damping profile for compliant control (we revisit the details of computing $\boldsymbol{K}_t$ in the next chapter), 4) set $j = j + 1$ and go to step 1 till the policy does not improve any more.

The overall algorithm combining the above stages is shown in Algorithm 3. We first learn the transition dynamics model of the environment from the available samples and use it to estimate the reward function for the locally optimal reference demonstrations. If the resulting control policy accumulates the same sum of rewards as that of human demonstrations, we conclude that the learned reward function, dynamics model and the control policy for the robot is optimal; otherwise we use the data samples generated by executing the control policy to improve the dynamics model and repeat the process again. Since we always execute the optimal policy on the robot for current estimate of the reward function and the dynamics model, we avoid any explicit exploration characteristic of model-free reinforcement learning approaches that may harm the robot.

Algorithm 3 *Model-Based IRL for Compliant Manipulation in Unknown Environment*

Input: Human demonstrations set $\mathcal{D}^{\pi_E}$

procedure `Continuous_IRL`

Initialize $\{\boldsymbol{\xi}_t, \boldsymbol{u}_t\}$ with few samples of the environment

i := 0

repeat

  i := i + 1

  Learn the dynamics model parameters with samples in $\{\boldsymbol{\xi}_t, \boldsymbol{u}_t\}$:

$$\dot{\boldsymbol{\xi}}_t = f_i(\boldsymbol{\xi}_t, \boldsymbol{u}_{t+1}; \theta)$$

  Find the reward function using $f_i(x_t, u_{t+1})$:

$$\boldsymbol{w}_i = \arg\max_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}|\mathcal{D}^{\pi_E})$$

  Compute the optimal policy using iLQR of the form:

$$\pi_i = \hat{\boldsymbol{u}}_t + \boldsymbol{K}_t(\hat{\boldsymbol{\xi}}_t - \boldsymbol{\xi}_t)$$

  Execute π_i on the real model and record samples

  Add samples $\{\boldsymbol{\xi}_t^{(m)}, \boldsymbol{u}_t^{(m)}\}_{t=1,m=1}^{T,M}$ to the set $\{\boldsymbol{\xi}_t, \boldsymbol{u}_t\}$

until $\|V^{\pi_i} - V^{\pi_E}\|_2$ is unchanged

return reward function, dynamics model, optimal policy

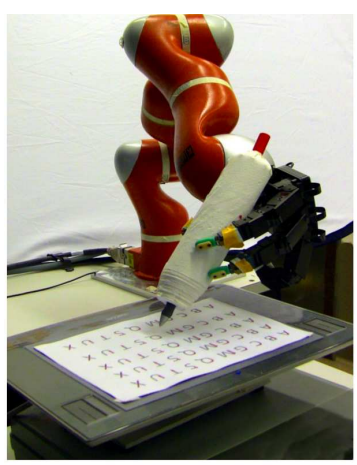


Figure 2.14: Letter writing setup with the KUKA robot.

2.4.2 Letter Writing Example

In this example, we are interested in learning compliant behaviour for the task of letter writing using human demonstrations. Writing a letter with a robot is a complex task that involves several key elements such as grasping force between the hand and the pen, tool-tip interaction force, orientation of the hand, etc [Yin 2016]. We verify our model-based IRL framework to find the reward function and write a letter in a compliant manner with the KUKA robot.

State-Action Space and Reward Function: The state of the robot is described by the Cartesian position of the pen mounted on the end-effector, $\boldsymbol{\xi}_t \in \mathbb{R}^3$, and the action corresponds to the 3−dimensional impedance force at the end-effector, $\boldsymbol{u}_t \in \mathbb{R}^3$. We denote the 3-dimensional impedance force with the tuple, $\{F_x, F_y, F_z\}$ to indicate the force in x,y and z-direction respectively. Let $\dot{\boldsymbol{\xi}}_t \in \mathbb{R}^3$ denote the tool-tip velocity of the end-effector. The dynamics model here maps the current tool-tip position and the impedance force to the tool-tip velocity which we represent as $\dot{\boldsymbol{\xi}}_t = f(\boldsymbol{\xi}_t, \boldsymbol{u}_t; \theta)$. We are interested in learning the control policy $\pi_t = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_T\}$ that regulates the interaction force with the environment over the course of writing a letter as observed in the human demonstrations.

The reward function is described by a trajectory following a given human demonstration for writing a letter (see Eq. (2.26)) with $\boldsymbol{Q} \in \mathbb{R}^{3\times3}$ and $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ as diagonal matrices containing the reward function parameters $\boldsymbol{w} \in \mathbb{R}^6$ with $\boldsymbol{Q} = \text{diag}\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}$, and $\boldsymbol{R} = \text{diag}\begin{bmatrix} w_4 & w_5 & w_6 \end{bmatrix}$.

Results and Discussions: We collect 2 trajectories of human demonstrations each for writing the letters $\{a, e, m, o, u, y\}$ using a tablet and measure the interaction forces with a six-axis force torque sensor placed below the tablet (see Fig. 2.14 for the experimental set-up). The demonstrations are used to learn the dynamics model, $\dot{\boldsymbol{\xi}}_t = f(\boldsymbol{\xi}_t, \boldsymbol{u}_t; \theta)$ using a total of 1000 data samples with SVR. The model predicts the Cartesian velocity given the Cartesian position of the end-effector and the impedance force. Parameter setting for SVR with $C = 100, \varepsilon = 0.01, \gamma = 0.5$, gives training and testing set error of 0.0036 and 0.0084 per datapoint. The learned model is used to estimate the reward function for which a given human demonstration is optimal. Learned trajectory reward function parameters for the dynamics model corresponds to $\boldsymbol{w} = \begin{bmatrix} 0.528 & 0.256 & 0.0004 & 0.267 & 0.732 & 0.216 \end{bmatrix}^\top$. The more the weight is along a particular dimension, the more is the required stiffness along that dimension. The optimal policy behavior is examined for the obtained reward function in Fig. 2.15. It can be seen that the variable stiffness feedback enables the robot to follow the desired trajectory of writing a letter under different noisy settings in a compliant manner.

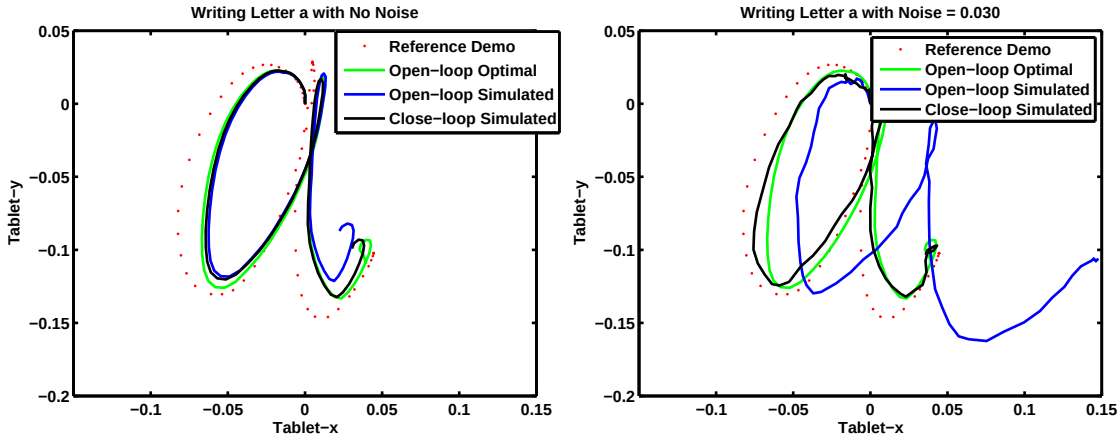


Figure 2.15: Optimal policy generalization with different noisy conditions for writing letter 'a': *(left)* no noise, *(right)* continuous noise added to state at each time step from a Gaussian distribution with mean 0 and standard deviation of 0.04.

2.5 Conclusions

This chapter presents a broad review of rewards-driven robot learning from demonstrations. In this paradigm, we first extract the reward function from the demonstrations to infer about the demonstrator preferences in a compact manner and then find the optimal policy to generalize better in different unobserved situations. We presented our approach of learning multiple reward functions in the demonstrations and used transfer learning to make IRL suitable for moderately high-dimensional spaces. To scale IRL in high-dimensional continuous domains, we moved from recursive value-function based methods to actor-critics with experience replay and policy gradient algorithms for finding the optimal policy in a model-free manner. To avoid explicit exploration during learning, we presented a model-based IRL approach in continuous unknown environments that iteratively updates the dynamic model, the reward function and the optimal policy to mimic the desired reference trajectory. In the next chapter, we revisit the problem of learning trajectory reward functions by performing statistics over the multiple demonstrations using generative models.

The rewards-driven paradigm is attractive for jointly addressing the what-to-imitate and how-to-imitate problem in learning from demonstrations, however, the practical issues in its implementation limit the widespread utility of IRL. Besides the high computational overhead of applying IRL in continuous unknown environments, a lot of effort goes into engineering the features (setting cut-off values, normalization range etc.) of the reward function whose unknown weighted combination should give the desired optimal policy. The

time required for iteratively designing the features is cumbersome to scale the paradigm for a wide range of problems. Despite the bottlenecks, IRL has been used in continuous domains to perform acrobatic helicopter manoeuvres [Abbeel 2010], humanoid locomotion [Mombaur 2010], playing table tennis [Muelling 2014], and grasping objects [Doerr 2015] among other applications. More recently, there has been a surge in applying deep variants of RL algorithms on robotic problems. Deep RL algorithms have been used to learn control policies from raw visual images [Levine 2015, Mnih 2015], and surpass humans in playing video games [Silver 2016]. Deep architectures for IRL provide a promising alternative to learn reward functions features with guided cost learning [Finn 2016], generative adversarial networks [Goodfellow 2014, Ho 2016], and unsupervised perceptual rewards [Sermanet 2016].

Chapter 3

Task-Parameterized Generative Models

Contents

3.1 Encoding with Generative Models 58

3.1.1 Gaussian Mixture Model (GMM) 58

3.1.2 Hidden Markov Model (HMM) 60

3.1.3 Hidden Semi-Markov Model (HSMM) 62

3.1.4 Kalman Filter and Dynamic Bayesian Networks 64

3.2 Task-Parameterized Generative Models 65

3.2.1 Learning Model Parameters 66

3.2.2 Adapting Model Parameters in New Situations 68

3.2.3 Sampling from HSMM 70

3.3 Decoding with Linear Quadratic Regulator/Tracking (LQR/LQT) 72

3.3.1 Continuous LQR/LQT 73

3.3.2 Discrete LQR/LQT 75

3.4 Valve Opening Example 76

3.5 Conclusion 78

Generative models are widely used to learn the distribution of the data for regenerating new samples from the model. Common examples include probability density function estimation, image regeneration and so on. Discriminative models, on the other hand, directly model the target variable(s) distribution given the observed variables. In this chapter, we learn the joint probability density function of the human demonstrations with

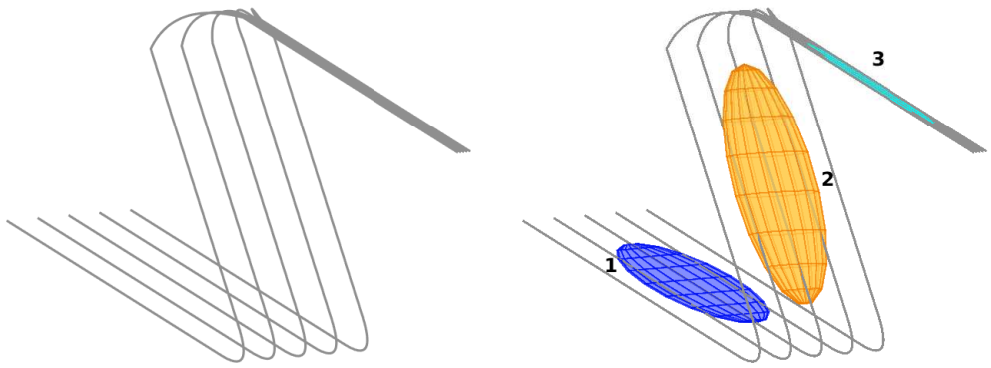


Figure 3.1: Encoding demonstrations on *(left)* with 3 components in a GMM on *(right)*.

a *hidden semi-Markov model* in an *unsupervised* manner, and smoothly follow the sampled sequence of states with a *linear quadratic tracking* controller. We show how the model can be systematically adopted to changing situations such as position/size/orientation of the objects in the environment with a *task-parameterized* formulation. We combine tools from statistical machine learning and optimal control to segment the demonstrations into different components or sub-goals that are sequenced together to perform manipulation tasks [Tanwani 2016a].

3.1 Encoding with Generative Models

3.1.1 Gaussian Mixture Model (GMM)

Probabilistic clustering models, such as Gaussian mixture model (GMM), are widely used to encode local trends in the data for classification or regression. For the set of T observations $\{\boldsymbol{\xi}_t\}_{t=1}^T$ with $\boldsymbol{\xi}_t \in \mathbb{R}^D$, the probability density function $\mathcal{P}$ of GMM with K mixture components is represented as,

$$\mathcal{P}(\boldsymbol{\xi}_t|\theta) = \sum_{i=1}^{K} \pi_i \, \mathcal{N}(\boldsymbol{\xi}_t|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \tag{3.1}$$

where $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ is the multivariate Gaussian distribution with prior π_i, mean $\boldsymbol{\mu}_i$, and covariance matrix $\boldsymbol{\Sigma}_i$. $\theta = \{\pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^K$ are the set of parameters to be estimated in the density function. In our case, we are mostly interested in clustering the data to encode their local trend based on the variance in the demonstrations, instead of trying to interpret the overall behaviour of the data. The observations $\{\boldsymbol{\xi}_t\}_{t=1}^T$ are assumed to be independent realizations of a random process whereas the unobserved labels $\{z_t\}_{t=1}^T$ are assumed to be

independent realizations of a random variable $z_t \in \{1, \ldots, K\}$. The set of pairs $\{\boldsymbol{\xi}_t, z_t\}_{t=1}^T$ compose the complete data set. The log-likelihood of the GMM is,

$$\mathcal{L}(\theta|\boldsymbol{\xi}) = \sum_{t=1}^{T} \log \left(\sum_{i=1}^{K} \pi_i \, \mathcal{N}(\boldsymbol{\xi}_t|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \right). \tag{3.2}$$

The inference of this model cannot be directly done through the maximization of the likelihood since the group labels $\{z_t\}_{t=1}^T$ of the observations are unknown. Given the initial set of parameters θ^{old}, the auxiliary function of GMM, $\mathcal{Q}(\theta, \theta^{\text{old}}) = \mathbb{E}\left\{\sum_{t=1}^T \log \mathcal{P}(\boldsymbol{\xi}_t, z_t|\theta) \mid \boldsymbol{\xi}_t, \theta^{\text{old}}\right\}$, takes the form [Dempster 1977],

$$\mathcal{Q}(\theta, \theta^{\text{old}}) \approx \frac{1}{2} \sum_{t=1}^{T} \sum_{i=1}^{K} h_{t,i}^{\theta^{\text{old}}} \left(\log \pi_i^2 - \log|\boldsymbol{\Sigma}_i| - (\boldsymbol{\xi}_t - \boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{\xi}_t - \boldsymbol{\mu}_i) - D \log(2\pi) \right), \tag{3.3}$$

where $h_{t,i}^{\theta^{\text{old}}} = p(z_t = i|\boldsymbol{\xi}_t, \theta^{\text{old}})$ is the probability of data point $\boldsymbol{\xi}_t$ to belong to i-th Gaussian component. Setting the derivative of the auxiliary function with respect to the model parameters equal to zero results in an Expectation-Maximization (EM) algorithm. The two steps are iteratively computed until the likelihood function converges to a local optimum.

E-step:

$$h_{t,i} = \frac{\pi_i \, \mathcal{N}(\boldsymbol{\xi}_t|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{k=1}^{K} \pi_k \, \mathcal{N}(\boldsymbol{\xi}_t|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}. \tag{3.4}$$

M-step:

$$\pi_i \leftarrow \frac{\sum_{t=1}^{T} h_{t,i}}{T}, \tag{3.5}$$

$$\boldsymbol{\mu}_i \leftarrow \frac{\sum_{t=1}^{T} h_{t,i} \, \boldsymbol{\xi}_t}{\sum_{t=1}^{T} h_{t,i}}, \tag{3.6}$$

$$\boldsymbol{\Sigma}_i \leftarrow \frac{\sum_{t=1}^{T} h_{t,i} \, (\boldsymbol{\xi}_t - \boldsymbol{\mu}_i)(\boldsymbol{\xi}_t - \boldsymbol{\mu}_i)^\top}{\sum_{t=1}^{T} h_{t,i}}. \tag{3.7}$$

Consider multiple demonstrations of a 3-dimensional Z-shaped movement as shown in Fig. 3.3. Encoding the demonstrations with 3 components in a GMM reveals two important aspects: 1) the mixture components identify sub-goals of a task, and 2) covariance in demonstrations gives a measure of the important aspects of a task, i.e., the less is the variance of a mixture component along a certain direction, the more constrained the model needs to be for reproduction along that direction.

3.1.2 Hidden Markov Model (HMM)

Hidden Markov models (HMMs) encapsulate the spatio-temporal information by augmenting a GMM with latent states that sequentially evolve over time in the demonstrations[1]. HMM is thus defined as a doubly stochastic process, one with sequence of hidden states and another with sequence of observations/emissions. Spatio-temporal encoding with HMMs can handle movements with variable durations, recurring patterns, options in the movement, or partial/unaligned demonstrations. HMMs are widely used for time series/sequence analysis in speech recognition, machine translation, DNA sequencing, robotics and many other fields [Rabiner 1989]. An interested reader can find more details about HMMs in [Rabiner 1989, Ghahramani 2002, Ephraim 2002].

Without loss of generality, we will describe the HMMs in which each state z_t is described by a single Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_{z_t}, \boldsymbol{\Sigma}_{z_t})$. In case of GMMs, the evaluation of each datapoint $\boldsymbol{\xi}_t$ is independent from the other datapoints in the stream of data. An HMM will instead consider transition probabilities between the K Gaussians, forming a $K \times K$ transition probability matrix, where an element $a_{i,j}$ in the matrix represents the probability to move from state i to state j in the next iteration. The parameters of an HMM will be described with parameters $\theta = \{\{a_{i,j}\}_{j=1}^{K}, \Pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{K}$, where Π_i are initial emission probabilities. For learning and inference in HMMs, it is useful to define intermediary variables, namely *forward variable* $\alpha_{t,i}^{\text{HMM}}$, *backward variable* $\beta_{t,i}^{\text{HMM}}$, *smoothed node marginal* $\boldsymbol{\gamma}_{t,i}^{\text{HMM}}$, and *smoothed edge marginal* $\zeta_{t,i,j}^{\text{HMM}}$.

Forward Variable - $\alpha_{t,i}^{\text{HMM}} \triangleq P(z_t = i, \boldsymbol{\xi}_1 \dots \boldsymbol{\xi}_t | \theta)$: The probability of a datapoint $\boldsymbol{\xi}_t$ to be in state i at time step t given the partial observation sequence $\{\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \dots, \boldsymbol{\xi}_t\}$, can be recursively computed with the forward variable as,

$$\alpha_{t,i}^{\text{HMM}} = \Big(\sum_{j=1}^{K} \alpha_{t-1,j}^{\text{HMM}} \, a_{j,i} \Big) \, \mathcal{N}(\boldsymbol{\xi}_t | \, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \tag{3.8}$$

with an initialization given by $\alpha_{1,i}^{\text{HMM}} = \Pi_i \, \mathcal{N}(\boldsymbol{\xi}_1 | \, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. Note that a naive computation would require marginalizing over all possible state sequences $\{z_1, \dots z_{t-1}\}$ which would grow exponentially with t. The forward variable takes advantage of the conditional independence in the network to perform the calculation recursively. Moreover, we consider HMMs with a single Gaussian as emission distribution, with $\pi_i = 1$, which is dropped in the equations. The forward variable can be used to compute the probability of the the full observation

[1] With a slight abuse of terminology, the word *state* is used to describe each discrete node/cluster encoding the evolution of a set of variables $\boldsymbol{\xi}_t$ in the context of HMM. It should not be confused with the word *state* in the context of dynamical systems that would instead be used to refer to the same set of variables $\boldsymbol{\xi}_t$.

$\boldsymbol{\xi} = \{\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_T\}$ with, $\mathcal{P}(\boldsymbol{\xi}|\theta) = \sum_{k=1}^{K} \alpha_{T,k}^{\text{HMM}}$.

Backward Variable - $\beta_{t,i}^{\text{HMM}} \triangleq P(\boldsymbol{\xi}_{t+1} \ldots \boldsymbol{\xi}_T | z_t = i, \theta)$: Similarly, a backward variable with a recursion going in the opposite direction can be defined by starting from $\beta_{T,i}^{\text{HMM}} = 1$ and by recursively computing,

$$\beta_{t,i}^{\text{HMM}} = \sum_{j=1}^{K} a_{i,j}\, \mathcal{N}(\boldsymbol{\xi}_{t+1} |\, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\, \beta_{t+1,j}^{\text{HMM}}, \tag{3.9}$$

which corresponds to the probability of the partial observation sequence $\{\boldsymbol{\xi}_{t+1}, \ldots, \boldsymbol{\xi}_{T-1}, \boldsymbol{\xi}_T\}$ given that we are in the i-th state at time step t.

Smoothed Node Marginal - $\boldsymbol{\gamma}_{t,i}^{\text{HMM}} \triangleq P(z_t = i | \boldsymbol{\xi}_1 \ldots \boldsymbol{\xi}_T, \theta)$: The probability of $\boldsymbol{\xi}_t$ to be in state i at time step t given the full observation sequence $\boldsymbol{\xi}$ is

$$\gamma_{t,i}^{\text{HMM}} = \frac{\alpha_{t,i}^{\text{HMM}} \beta_{t,i}^{\text{HMM}}}{\sum\limits_{k=1}^{K} \alpha_{t,k}^{\text{HMM}} \beta_{t,k}^{\text{HMM}}} = \frac{\alpha_{t,i}^{\text{HMM}} \beta_{t,i}^{\text{HMM}}}{\mathcal{P}(\boldsymbol{\xi}|\theta)}. \tag{3.10}$$

Smoothed Edge Marginal - $\zeta_{t,i,j}^{\text{HMM}} \triangleq P(z_t = i, z_{t+1} = j | \boldsymbol{\xi}_1 \ldots \boldsymbol{\xi}_T, \theta)$: The probability of $\boldsymbol{\xi}_t$ to be in state i at time step t and in state j at time step $t+1$ given the full observation sequence $\boldsymbol{\xi}$ is

$$\zeta_{t,i,j}^{\text{HMM}} = \frac{\alpha_{t,i}^{\text{HMM}}\, a_{i,j}\, \mathcal{N}(\boldsymbol{\xi}_{t+1} |\, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\, \beta_{t+1,j}^{\text{HMM}}}{\sum\limits_{k=1}^{K} \sum\limits_{l=1}^{K} \alpha_{t,k}^{\text{HMM}}\, a_{k,l}\, \mathcal{N}(\boldsymbol{\xi}_{t+1} |\, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)\, \beta_{t+1,l}^{\text{HMM}}} = \frac{\alpha_{t,i}^{\text{HMM}}\, a_{i,j}\, \mathcal{N}(\boldsymbol{\xi}_{t+1} |\, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\, \beta_{t+1,j}^{\text{HMM}}}{\mathcal{P}(\boldsymbol{\xi}|\theta)}. \tag{3.11}$$

The expected complete log-likelihood of HMMs for a set of M demonstrations defined with an additional index m, $\mathcal{Q}(\theta, \theta^{\text{ old}}) = \mathbb{E}\left\{\sum_{m=1}^{M} \sum_{t=1}^{T} \log \mathcal{P}(\boldsymbol{\xi}_{m,t}, z_t | \theta) \mid \boldsymbol{\xi}, \theta^{\text{ old}}\right\}$, is given as

$$\mathcal{Q}(\theta, \theta^{\text{ old}}) = \sum_{i=1}^{K} \sum_{m=1}^{M} \gamma_{m,1,i}^{\text{HMM}} \log \Pi_i \; + \sum_{i=1}^{K} \sum_{j=1}^{K} \sum_{m=1}^{M} \sum_{t=1}^{T} \zeta_{m,t,i,j}^{\text{HMM}} \log a_{i,j} \; + \sum_{m=1}^{M} \sum_{t=1}^{T} \sum_{i=1}^{K} \mathcal{P}(z_t = i | \boldsymbol{\xi}_{m,t}, \theta^{\text{old}}) \log \mathcal{N}(\boldsymbol{\xi}_{m,t} | \boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}). \tag{3.12}$$

Maximizing $\mathcal{Q}(\theta, \theta^{\text{ old}})$ with respect to the model parameters θ for iteratively performing the EM steps with

E-step:

$$\gamma^{\text{HMM}}_{m,t,i} = \frac{\alpha^{\text{HMM}}_{t,i}\beta^{\text{HMM}}_{t,i}}{\sum\limits_{k=1}^{K}\alpha^{\text{HMM}}_{t,k}\beta^{\text{HMM}}_{t,k}}, \tag{3.13}$$

M-step:

$$\Pi_i \leftarrow \frac{\sum_{m=1}^{M}\gamma^{\text{HMM}}_{m,1,i}}{M}, \tag{3.14}$$

$$a_{i,j} \leftarrow \frac{\sum_{m=1}^{M}\sum_{t=1}^{T_m-1}\zeta^{\text{HMM}}_{m,t,i,j}}{\sum_{m=1}^{M}\sum_{t=1}^{T_m-1}\gamma^{\text{HMM}}_{m,t,i}}, \tag{3.15}$$

$$\boldsymbol{\mu}_i \leftarrow \frac{\sum_{m=1}^{M}\sum_{t=1}^{T_m}\gamma^{\text{HMM}}_{m,t,i}\,\boldsymbol{\xi}_{m,t}}{\sum_{m=1}^{M}\sum_{t=1}^{T_m}\gamma^{\text{HMM}}_{m,t,i}}, \tag{3.16}$$

$$\boldsymbol{\Sigma}_i \leftarrow \frac{\sum_{m=1}^{M}\sum_{t=1}^{T_m}\gamma^{\text{HMM}}_{m,t,i}(\boldsymbol{\xi}_{m,t}-\boldsymbol{\mu}_i)(\boldsymbol{\xi}_{m,t}-\boldsymbol{\mu}_i)^\top}{\sum_{m=1}^{M}\sum_{t=1}^{T_m}\gamma^{\text{HMM}}_{m,t,i}}. \tag{3.17}$$

Note that numerical underflow issues easily occur with a naive implementation of the above algorithms. In practice, a simple approach to avoid this issue is to rely on scaling factors during the computation of the forward and backward variables, which get canceled out when normalizing the posterior [Rabiner 1989].

3.1.3 Hidden Semi-Markov Model (HSMM)

Semi-Markov models relax the Markovian structure of state transitions by relying not only upon the current state but also on the duration/elapsed time in the current state, i.e., the underlying process is defined by a *semi-Markov* chain with a variable duration time for each state. The state duration stay is a random integer variable that assumes values in the set $\{1, 2, \ldots, s^{\max}\}$. The value corresponds to the number of observations produced in a given state, before transitioning to the next state. The parameter set now additionally contains the duration distribution for each state p_i^s and the transition probability also becomes dependent on the duration of time spent in the previous state $s^{'}$ and the current state s, denoted as $a_{(i,s')(j,s)}$. **Hidden Semi-Markov Models** (HSMMs) associate an observable output distribution with each state in a semi-Markov chain, similar to how we associated a sequence of observations with a Markov chain in a HMM [Yu 2010]. Depending upon the assumptions on transition from one state to the other, HSMMs are classified into different groups including,

- Non-stationary HSMMs in [Marhasev 2006] were introduced in which the state

duration may be assumed to be independent to the previous state specified as $a_{(i,s')(j,s)} = a_{(i,s')\,j}\ p_j^s$, i.e., the state switches to state j for the probability duration of s steps after spending $s^{'}$ duration steps in state i.

- In **residential time HSMMs** [Yu 2006], the state transition is assumed to be independent to the duration of the previous state with $a_{(i,s')(j,s)} = a_{i\,(j,s)}$ specifying the transition probability that state i ends at time $t-1$ and transits to state j for s duration steps.

- In **variable transition HSMMs** [Krishnamurthy 1991, Ramesh 1992], the self-transition is allowed and assumed to be independent to the previous state with $a_{(i,s')(j,s)} = a_{(i,s')\,j}\ \prod_{\tau=1}^{s-1} a_{jj}(\tau)[1-a_{jj}(s)]$, where $a_{jj}(s)$ is the self-transition probability when state j has stayed for duration s, and $1-a_{jj}(s)$ is the probability that state j ends with duration s. The model is realized as a HMM with augmented state (i,s).

- In **explicit duration HSMMs** [Yu 2006], the transition to the current state is independent to the duration of the previous state and the duration is only dependent on the current state with $a_{(i,s')(j,s)} = a_{i,j}\ p_j^s$. The transition probability specifies the switch from state i at time t to state j for a stay of duration of s steps in state j at time $[t+1, t+d]$. The self transition probabilities $a_{i,i}$ are set to zero in the explicit duration model. The scope of this thesis is limited to explicit duration HSMMs.

Though HMMs also implicitly assume that the duration of staying in a state follows a geometric distribution, this assumption is often limiting, especially for modelling the sequences with long state dwell times [Rabiner 1989]. The probability of staying s consecutive time steps in state i exponentially decreases with time in HMMs as

$$p_i^s = a_{i,i}^{s-1}(1-a_{i,i}). \tag{3.18}$$

An explicit duration HSMM sets the self-transition probabilities to zero and explicitly models the state duration with a parametric distribution. We use a Gaussian distribution here to model the state duration with parameters $\{\mu_i^S, \Sigma_i^S\}$. Note that a lognormal, gamma or a poisson distribution may also be used to avoid sampling negative time duration steps from the Gaussian distribution, but this effect is most often negligible in robotic applications[2]. The parameter set for an HSMM is defined by $\left\{\Pi_i, \{a_{i,m}\}_{m=1}^K, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \mu_i^S, \Sigma_i^S\right\}_{i=1}^K$. Learning and inference in HSMMs requires computing the intermediary variables as defined for the case of HMMs. In practice, as an approximation, we adopted a simpler approach

[2]We encode movements with few states and a duration distribution whose center is most often far from zero with relatively small variance.

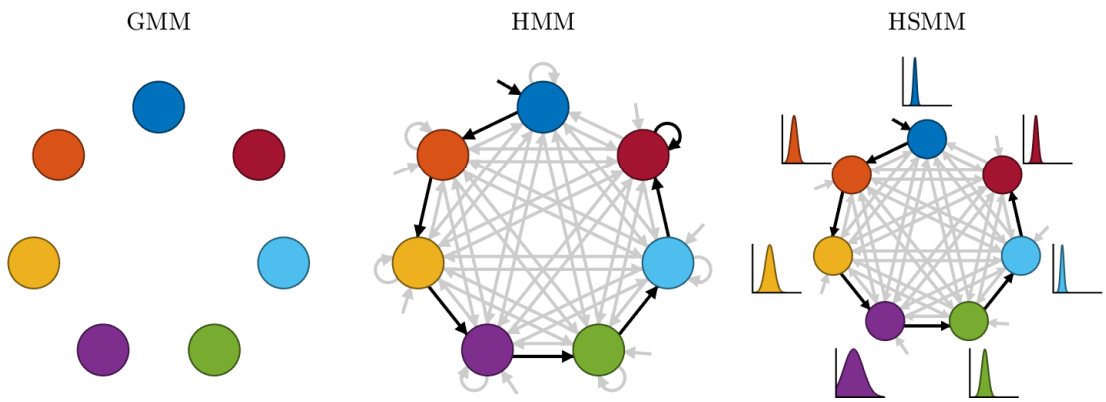


Figure 3.2: Graphical representation of a GMM, HMM and HSMM with 7 components (see [Calinon 2011, Tanwani 2016a, Tanwani 2016c] for use in robotic applications).

to learn the model parameters. Parameters $\left\{\Pi_i, \{a_{i,m}\}_{m=1}^K, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right\}_{i=1}^K$ are estimated using the EM algorithm for HMMs as described in the previous section, and the duration parameters $\{\mu_i^S, \Sigma_i^S\}_{i=1}^K$ are estimated empirically from the data after training from the most likely hidden state sequence $\boldsymbol{z}_t = \{z_1 \dots z_T\}$. In Fig. 3.2, we provide a graphical representation of the difference of encoding among GMM, HMM and a HSMM. A GMM model encodes the structure of the motion but does not model the transition between the states. An HMM uses transition probabilities and self-transition probabilities to switch among states. Self-state transitions are known to only poorly describe the probability that the system is expected to stay in a given state for a long duration. The HSMM model instead explicitly models the state duration probabilities as Gaussian distributions, while keeping the transition probabilities across states.

3.1.4 Kalman Filter and Dynamic Bayesian Networks

It is useful to note that several other modelling representations can be considered to encode the spatio-temporal patterns in the observations. For example, a Kalman filter model represents the transition distribution between latent states in a HMM with a continuous linear dynamical system or a linear state space model, i.e.,

$$\mathcal{P}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) = \mathcal{N}(\boldsymbol{z}_t; \boldsymbol{A}_k\boldsymbol{z}_{t-1} + \boldsymbol{\mu}_z, \boldsymbol{Q}_z), \tag{3.19}$$

$$\mathcal{P}(\boldsymbol{\xi}_t|\boldsymbol{z}_t) = \mathcal{N}(\boldsymbol{\xi}_t; \boldsymbol{C}_k\boldsymbol{\xi}_t + \boldsymbol{\mu}_\xi, \boldsymbol{Q}_\xi), \tag{3.20}$$

where $\boldsymbol{z}_t \in \mathbb{R}^{D_z}$ is a continuous D_z dimensional latent variable, $\boldsymbol{A}_k, \boldsymbol{C}_k$ are the linear transformation matrices for hidden state and observation respectively, $\boldsymbol{\mu}_z, \boldsymbol{\mu}_\xi$ are the inde-

pendent additive noise mean variables of state and observation respectively, while $\boldsymbol{Q}_z, \boldsymbol{Q}_\xi$ are the independent positive semi-definite matrices of state and observation noise respectively. The inference and learning in Kalman filter with linear Gaussian assumptions is done in an exact manner, while Extended Kalman filter or Unscented Kalman filter are used for approximating non-linear dynamics [Wan 2000].

Both HMMs and Kalman filter can be considered as special cases of Dynamic Bayesian Networks (DBNs), that can model complex patterns in sequential data at the cost of increased computational and algorithmic complexity. Other important variants of DBNs include auto-regressive models, input-output HMMs, factorial HMMs, hierarchical HMMs, abstract HMMs. An interested reader can find more details of these variants in [Murphy 2012].

3.2 Task-Parameterized Generative Models

With increasing functional and behavioural expectations of robots, it has become imperative to encode manipulation tasks such that the robots are able to execute them in previously unseen contexts. It is often difficult to collect a set of demonstrations for all possible situations and operating conditions of the task. The reproduction phase also faces a similar issue, i.e., after having observed a set of demonstrations in some situations, we would like to generalize the skill to new situations. Task-parameterized models provide a probabilistic formulation to deal with different real world situations by adapting the model parameters [Wilson 1999, Yamazaki 2005, Krueger 2010, Ureche 2015, Yang 2015, Silverio 2015, Calinon 2016, Tanwani 2016a], instead of hard coding the solution for each new situation or handling it in an ad hoc manner.

Most of the existing methods retrieve the movement from the model parameters and the task parameters as a standard regression problem [Inamura 2004, Alissandrakis 2006, Ude 2010, Kronander 2011, Kober 2012, Mühlig 2012, Paraschos 2013]. This generality might look appealing at first sight, but it also strongly limits and bounds the generalization scope of these models. Task-parameterized models handle new environmental situations by defining external coordinate systems or frames of reference[3]. For example, a coordinate system can be attached to an object whose position and orientation may change during the task. When a different situation occurs (position/orientation of the object changes), changes in the task parameters or reference frames are used to modulate the model parameters in order to adapt the robot movement to the new situation. We denote *task parameters* to refer to the coordinate systems that describe the current environmental situation, such

[3]We use the term coordinate system and frame of reference interchangeably in this thesis.

as positions of objects in the environment. The *model parameters* refer to the parameters learned by the system to encode the movement.

3.2.1 Learning Model Parameters

We represent the task parameters with P coordinate systems, defined by the coordinate systems $\{\boldsymbol{A}_j, \boldsymbol{b}_j\}_{j=1}^P$, where $\boldsymbol{A}_j$ denotes the orientation of the coordinate system as a rotation matrix and $\boldsymbol{b_j}$ represents the origin of the coordinate system.[4] The observations $\boldsymbol{\xi}_t$ are observed from different coordinate systems forming a third order tensor dataset $\{\boldsymbol{\xi}_t^{(j)}\}_{t,j=1}^{T,P}$ with

$$\boldsymbol{\xi}_t^{(j)} = \boldsymbol{A}_j^{-1}(\boldsymbol{\xi}_t - \boldsymbol{b}_j). \tag{3.21}$$

3.2.1.1 Task-Parameterized GMM

The parameters of the task-parameterized GMM are defined by $\theta_p = \{\pi_i, \{\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\}_{j=1}^P\}_{i=1}^K$, where $\boldsymbol{\mu}_i^{(j)}$ and $\boldsymbol{\Sigma}_i^{(j)}$ define the mean and the covariance matrix of the i-th mixture component in frame P. Learning of the parameters is achieved with the constrained problem of maximizing the log-likelihood under the constraints that the data in the different frames are generated from the same source, resulting in an EM process to iteratively update the model parameters until convergence. The probability of data point $\boldsymbol{\xi}_t$ to belong to the i-th Gaussian component at time t (E-step) in the task-parameterized formulation is given by [Calinon 2016]

E-step:

$$h_{t,i}^{\hat{\theta}_p} = \frac{\pi_i \prod_{j=1}^{P} \mathcal{N}(\boldsymbol{\xi}_t^{(j)} | \, \boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)})}{\sum_{k=1}^K \pi_k \prod_{j=1}^{P} \mathcal{N}(\boldsymbol{\xi}_t^{(j)} | \, \boldsymbol{\mu}_k^{(j)}, \boldsymbol{\Sigma}_k^{(j)})}, \tag{3.22}$$

where $\prod_{j=1}^{P} \mathcal{N}(\boldsymbol{\xi}_t^{(j)} | \, \boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)})$ represents the product of the probabilities of the datapoint observed in P frames to belong to i-th Gaussian in the corresponding frame. Maximum likelihood estimates of the parameters remain the same as in a GMM except the computation is repeated with respect to P different frames, i.e.,

[4] Without loss of generality, the frames can be time-varying defined at time t by $\{\boldsymbol{A}_{t,j}, \boldsymbol{b}_{t,j}\}_{j=1}^P$.

M-step:

$$\pi_i \leftarrow \frac{\sum_{t=1}^{T} h_{t,i}}{T}, \tag{3.23}$$

$$\boldsymbol{\mu}_i^{(j)} \leftarrow \frac{\sum_{t=1}^{T} h_{t,i}\, \boldsymbol{\xi}_t^{(j)}}{\sum_{t=1}^{T} h_{t,i}}, \tag{3.24}$$

$$\boldsymbol{\Sigma}_i^{(j)} \leftarrow \frac{\sum_{t=1}^{T} h_{t,i}\, (\boldsymbol{\xi}_t^{(j)} - \boldsymbol{\mu}_i^{(j)})(\boldsymbol{\xi}_t^{(j)} - \boldsymbol{\mu}_i^{(j)})^\top}{\sum_{t=1}^{T} h_{t,i}}. \tag{3.25}$$

3.2.1.2 Task-Parameterized HSMM

In order to learn the HSMM in the task-parameterized formulation, we assume that the emission distribution of the i-th state is represented by the product of the probabilities of the datapoint observed in P frames to belong to the i-th Gaussian in the corresponding coordinate system. The **forward variable** of HMM in the task-parameterized formulation is described as

$$\alpha_{t,i}^{\text{TP-HMM}} = \Big(\sum_{j=1}^{K} \alpha_{t-1,j}^{\text{HMM}}\, a_{j,i}\Big) \prod_{j=1}^{P} \mathcal{N}(\boldsymbol{\xi}_t^{(j)} |\, \boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}). \tag{3.26}$$

Similarly, the **backward variable** $\beta_{t,i}^{\text{TP-HMM}}$, the **smoothed node marginal** $\gamma_{t,i}^{\text{TP-HMM}}$, and the **smoothed edge marginal** $\zeta_{t,i,j}^{\text{TP-HMM}}$ can be computed by replacing the emission distribution $\mathcal{N}(\boldsymbol{\xi}_t |\, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ in Eq. (3.9), Eq. (3.10) and Eq. (3.11) with the product of probabilities of the datapoint in each frame $\prod_{j=1}^{P} \mathcal{N}(\boldsymbol{\xi}_t^{(j)} |\, \boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)})$.

The initial state probability Π_i and the transition probability $a_{i,m}$ of moving to state m are represented in the same manner as that of HSMM. The parameters of task-parameterized HSMM are described by $\theta_h = \Big\{\Pi_i, \{a_{i,m}\}_{m=1}^{K}, \{\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\}_{j=1}^{P}, \mu_i^S, \Sigma_i^S\Big\}_{i=1}^{K}$. Parameters $\Big\{\Pi_i, \{a_{i,m}\}_{m=1}^{K}, \{\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\}_{j=1}^{P}\Big\}_{i=1}^{K}$ are estimated with intermediary variables defined as above using EM in an iterative manner, while the parameters $\{\mu_i^S, \Sigma_i^S\}_{i=1}^{K}$ are estimated empirically from the data after training similar to the HSMM case. The resulting EM steps are summarized as [Tanwani 2016a]

E-step:

$$\gamma_{m,t,i}^{\text{TP-HMM}} = \frac{\alpha_{t,i}^{\text{TP-HMM}} \beta_{t,i}^{\text{TP-HMM}}}{\sum\limits_{k=1}^{K} \alpha_{t,k}^{\text{TP-HMM}} \beta_{t,k}^{\text{TP-HMM}}}, \tag{3.27}$$

M-step:

$$\Pi_i \leftarrow \frac{\sum_{m=1}^{M} \gamma_{m,1,i}^{\text{TP-HMM}}}{M}, \tag{3.28}$$

$$a_{i,j} \leftarrow \frac{\sum_{m=1}^{M} \sum_{t=1}^{T_m-1} \zeta_{m,t,i,j}^{\text{TP-HMM}}}{\sum_{m=1}^{M} \sum_{t=1}^{T_m-1} \gamma_{m,t,i}^{\text{TP-HMM}}}, \qquad (i \neq j) \tag{3.29}$$

$$\boldsymbol{\mu}_i^{(j)} \leftarrow \frac{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma_{m,t,i}^{\text{TP-HMM}} \boldsymbol{\xi}_{m,t}^{(j)}}{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma_{m,t,i}^{\text{TP-HMM}}}, \tag{3.30}$$

$$\boldsymbol{\Sigma}_i^{(j)} \leftarrow \frac{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma_{m,t,i}^{\text{TP-HMM}} (\boldsymbol{\xi}_{m,t}^{(j)} - \boldsymbol{\mu}_i^{(j)})(\boldsymbol{\xi}_{m,t}^{(j)} - \boldsymbol{\mu}_i^{(j)})^\top}{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma_{m,t,i}^{\text{TP-HMM}}}. \tag{3.31}$$

The duration model $\mathcal{N}(s|\mu_i^S, \Sigma_i^S)$ is used as a replacement of the self-transition probabilities $a_{i,i}$. The hidden state sequence of the demonstrations is obtained as,

$$z_t = \arg\max_i \ \gamma_{m,t,i}^{\text{TP-HMM}}. \tag{3.32}$$

The hidden state sequence over all demonstrations is used to define the duration model parameters $\{\mu_i^S, \Sigma_i^S\}$ as the mean and the standard deviation of staying s consecutive time steps in the i-th state.

3.2.2 Adapting Model Parameters in New Situations

In order to define the model parameters in new situations, we make use of two properties of multivariate Gaussians:

Linear Transformation of Gaussians: If $\boldsymbol{\xi}_t$ follows Gaussian distribution $\mathcal{N}(\boldsymbol{u}, \boldsymbol{\Sigma})$, then the linear transformation of the data $\boldsymbol{A}\boldsymbol{\xi}_t + \boldsymbol{b}$ in the coordinate system $\{\boldsymbol{A}, \boldsymbol{b}\}$ follows the distribution

$$\boldsymbol{A}\boldsymbol{\xi}_t + \boldsymbol{b} \sim \mathcal{N}(\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{b}, \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\top). \tag{3.33}$$

Product of Gaussians: The product of two multivariate Gaussians $\mathcal{N}(\boldsymbol{\mu}^{(1)}, \boldsymbol{\Sigma}^{(1)})$ and $\mathcal{N}(\boldsymbol{\mu}^{(2)}, \boldsymbol{\Sigma}^{(2)})$ can be approximated as a multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu}^p, \boldsymbol{\Sigma}^p)$ with

$$\begin{aligned} \mathcal{N}(\boldsymbol{\mu}^p, \boldsymbol{\Sigma}^p) &\propto \mathcal{N}(\boldsymbol{\mu}^{(1)}, \boldsymbol{\Sigma}^{(1)}) \cdot \mathcal{N}(\boldsymbol{\mu}^{(2)}, \boldsymbol{\Sigma}^{(2)}), \\ \text{where,} \quad \boldsymbol{\Sigma}^p &= \left(\boldsymbol{\Sigma}^{(1)} + \boldsymbol{\Sigma}^{(2)}\right)^{-1}, \\ \boldsymbol{\mu}^p &= \boldsymbol{\Sigma}^p \left(\boldsymbol{\Sigma}^{(1)^{-1}} \boldsymbol{\mu}^{(1)} + \boldsymbol{\Sigma}^{(2)^{-1}} \boldsymbol{\mu}^{(2)}\right). \end{aligned} \tag{3.34}$$

Intuitively speaking, the product of Gaussians gives a closed form expression for minimizing

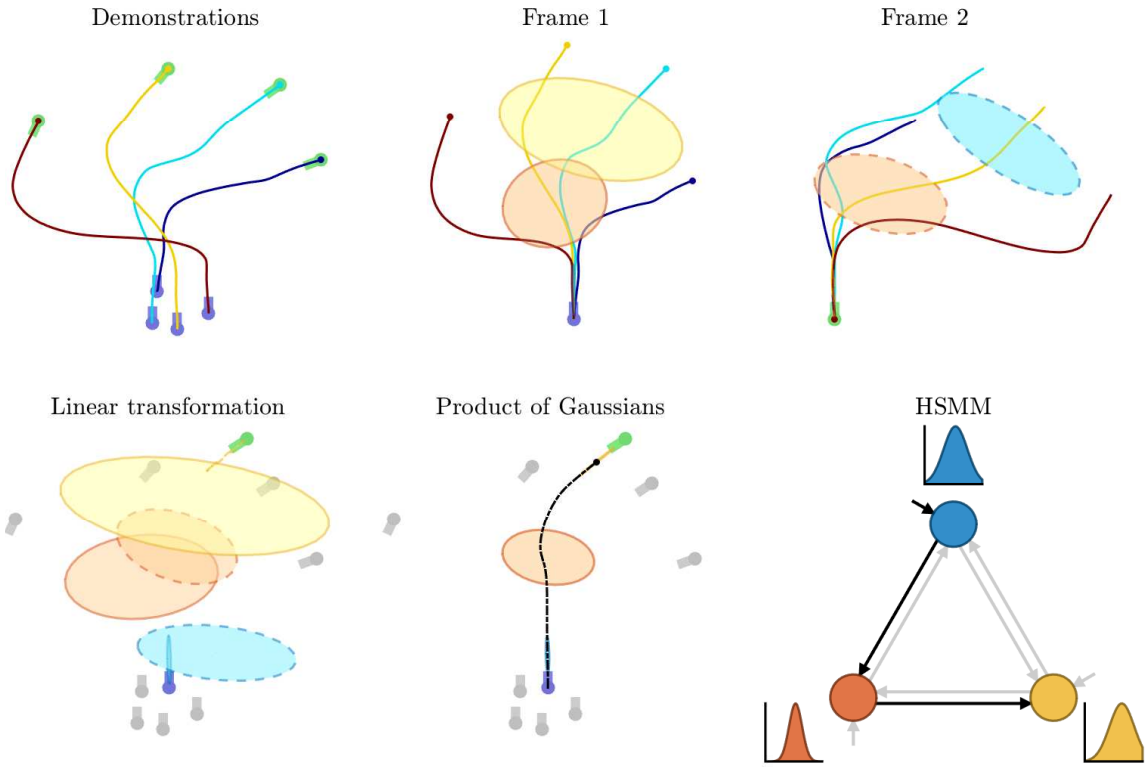


Figure 3.3: Task-parameterized HSMM: *(top-left)* each demonstration is observed with respect to frame 1 (in purple) and frame 2 (in green) respectively, *(top-center, top-right)* model is learned in respective coordinate systems, *(bottom-left)* linear transformation of Gaussians for new environmental situation described by coordinate systems in purple and in green, *(bottom-center)* product of linearly transformed Gaussians and movement reproduction for a new situation, *(bottom-right)* HSMM graphical representation.

the cost on the resulting $\boldsymbol{\mu}^p$,

$$\boldsymbol{\mu}^p = \arg\min_{\boldsymbol{\xi}} \sum_{i=1}^{2} \left(\boldsymbol{\xi} - \boldsymbol{\mu}^{(i)}\right)^\top \boldsymbol{\Sigma}^{(i)^{-1}} \left(\boldsymbol{\xi} - \boldsymbol{\mu}^{(i)}\right), \tag{3.35}$$

with a minimization error given by $\boldsymbol{\Sigma}^p$. The product of P multivariate Gaussians follows essentially the same principle. Using these two properties, we adapt the model parameters for a new unseen environmental situation described by the frames $\{\tilde{\boldsymbol{A}}_j, \tilde{\boldsymbol{b}}_j\}_{j=1}^P$ after the training phase. The new model parameters $\{\tilde{\boldsymbol{\mu}}_i, \tilde{\boldsymbol{\Sigma}}_i\}$ for the i-th mixture component correspond to the product of the linearly transformed i-th Gaussian components in P frames

$$\mathcal{N}(\tilde{\boldsymbol{\mu}}_i, \tilde{\boldsymbol{\Sigma}}_i) \propto \prod_{j=1}^{P} \mathcal{N}\left(\tilde{\boldsymbol{A}}_j \boldsymbol{\mu}_i^{(j)} + \tilde{\boldsymbol{b}}_j, \tilde{\boldsymbol{A}}_j \boldsymbol{\Sigma}_i^{(j)} \tilde{\boldsymbol{A}}_j^\top\right). \tag{3.36}$$

Evaluating the product of Gaussian yields

$$\begin{aligned} \tilde{\boldsymbol{\Sigma}}_i &= \left(\sum_{j=1}^{P} \left(\tilde{\boldsymbol{A}}_j \boldsymbol{\Sigma}_i^{(j)} \tilde{\boldsymbol{A}}_j^\top\right)^{-1}\right)^{-1}, \\ \tilde{\boldsymbol{\mu}}_i &= \tilde{\boldsymbol{\Sigma}}_i \sum_{j=1}^{P} \left(\tilde{\boldsymbol{A}}_j \boldsymbol{\Sigma}_i^{(j)} \tilde{\boldsymbol{A}}_j^\top\right)^{-1} \left(\tilde{\boldsymbol{A}}_j \boldsymbol{\mu}_i^{(j)} + \tilde{\boldsymbol{b}}_j\right). \end{aligned} \tag{3.37}$$

3.2.3 Sampling from HSMM

Given the new model parameters $\{\tilde{\boldsymbol{\mu}}_i, \tilde{\boldsymbol{\Sigma}}_i\}_{i=1}^K$ and a sequence of observations $\{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_t\}$, we are interested in predicting the probability of the hidden state sequence over the next time horizon T_p, i.e., $p(z_t, z_{t+1}, \ldots, z_{T_p} \mid \boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_t)$ [Tanwani 2016a].

Starting from the initial datapoint $\boldsymbol{\xi}_1$, the probability of datapoint to belong to the i-th mixture component is

$$h_{1,i}^{\text{HMM}} = \frac{\pi_i \mathcal{N}(\boldsymbol{\xi}_1 | \tilde{\boldsymbol{\mu}}_i, \tilde{\boldsymbol{\Sigma}}_i)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{\xi}_1 | \tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k)}. \tag{3.38}$$

The probability of the observed sequence $\{\boldsymbol{\xi}_1 \ldots \boldsymbol{\xi}_t\}$ to belong to a hidden state $z_t = i$ at the end of the sequence (also known as *filtering* problem) is computed with the help of the forward variable as

$$p(z_t \mid \boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_t) = h_{t,i}^{\text{HMM}} = \frac{\alpha_{t,i}^{\text{HMM}}}{\sum_{k=1}^{K} \alpha_{t,k}^{\text{HMM}}}. \tag{3.39}$$

Sampling from the model for predicting the sequence of states over the next time horizon

can take two forms:

1) Stochastic Sampling: The sequence of states is sampled in a probabilistic manner given the state duration and the state transition probabilities. By stochastic sampling, motions that contain different options and do not evolve only on a single path can also be represented (see [Gowrishankar 2013] for example). The procedure is as follows:

1. start from the initial state $z_t = i$,
2. sample the s duration steps from $\{\mu_i^S, \Sigma_i^S\}$,
3. sample the next transition state $z_{t+s+1} \sim \boldsymbol{\pi}_{z_{t+s}}$ from the transition probability matrix,
4. repeat step 2 and step 3 until T_p duration steps.

2) Deterministic Sampling: The most likely sequence of states is sampled and remains unchanged in successive sampling trials. We use the forward variable of HSMM for deterministic sampling from the model. The forward variable $\alpha_{t,i}^{\text{HSMM}} \triangleq P(z_t = i, \boldsymbol{\xi}_1 \dots \boldsymbol{\xi}_t | \theta)$ requires marginalizing over the duration steps along with all possible state sequences. The probability of a datapoint $\boldsymbol{\xi}_t$ to be in state i at time step t given the partial observation sequence $\{\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \dots, \boldsymbol{\xi}_t\}$ is now specified as [Yu 2010]

$$\alpha_{t,i}^{\text{HSMM}} = \sum_{s=1}^{\min(s^{\max},t-1)} \sum_{j=1}^{K} \alpha_{t-s,j}^{\text{HSMM}} \, a_{j,i} \, \mathcal{N}(s|\mu_i^S, \Sigma_i^S) \prod_{c=t-s+1}^{t} \mathcal{N}(\boldsymbol{\xi}_c | \, \tilde{\boldsymbol{\mu}}_i, \tilde{\boldsymbol{\Sigma}}_i), \tag{3.40}$$

where the initialization is given by $\alpha_{1,i}^{\text{HSMM}} = \Pi_i \; \mathcal{N}(1|\mu_i^S, \Sigma_i^S) \; \mathcal{N}(\boldsymbol{\xi}_1 | \; \tilde{\boldsymbol{\mu}}_i, \tilde{\boldsymbol{\Sigma}}_i)$, and the output distribution in state i is conditionally independent for the s duration steps given as $\prod_{c=t-s+1}^{t} \mathcal{N}(\boldsymbol{\xi}_c | \, \tilde{\boldsymbol{\mu}}_i, \tilde{\boldsymbol{\Sigma}}_i)$. Note that for $t < s^{\max}$, the sum over duration steps is computed for $t-1$ steps, instead of $s^{\max}$. Without the observation sequence for the next time steps, the forward variable simplifies to

$$\alpha_{t,i}^{\text{HSMM}} = \sum_{s=1}^{\min(s^{\max},t-1)} \sum_{j=1}^{K} \alpha_{t-s,j}^{\text{HSMM}} \, a_{j,i} \, \mathcal{N}(s|\mu_i^S, \Sigma_i^S). \tag{3.41}$$

The forward variable is used to plan the movement sequence for the next T_p steps with $t = t+1 \dots T_p$. During prediction, we only use the transition matrix and the duration model to plan the future evolution of the initial/current state and omit the influence of the spatial data that we cannot observe, i.e., $\mathcal{N}(\boldsymbol{\xi}_t | \tilde{\boldsymbol{\mu}}_i, \tilde{\boldsymbol{\Sigma}}_i) = 1$ for $t > 1$. This is used to retrieve a step-wise reference trajectory $\mathcal{N}(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t)$ from a given state sequence $\boldsymbol{z}_t$ computed

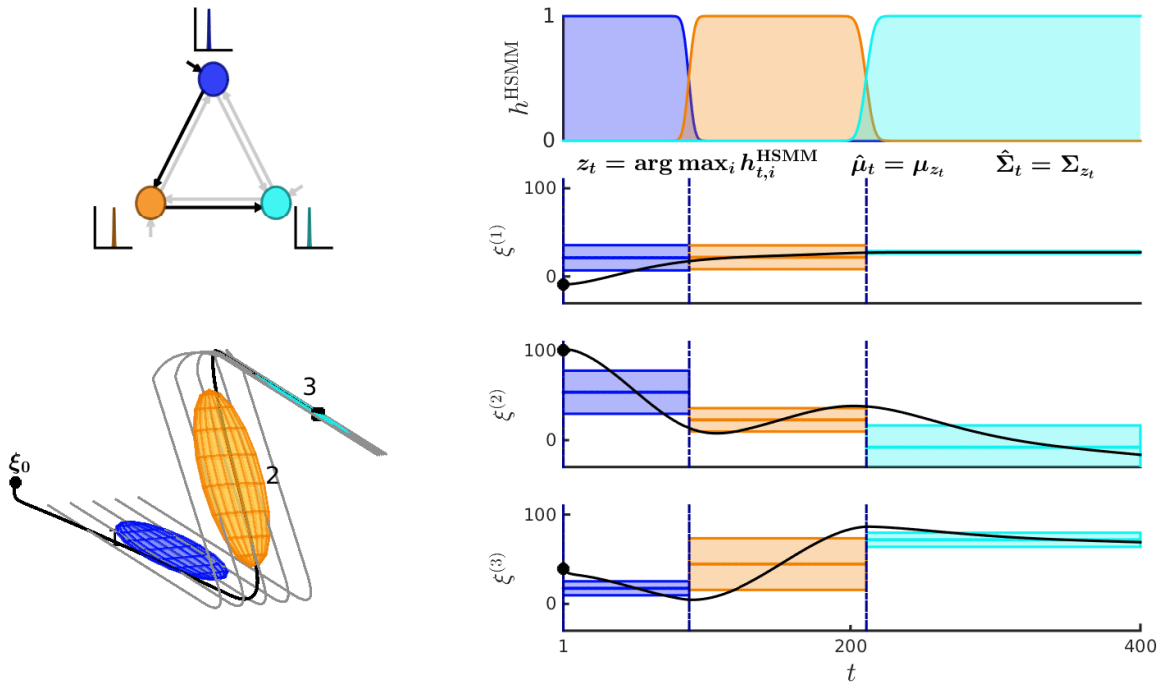


Figure 3.4: Sampling from HSMM in the initial state $\boldsymbol{\xi}_0$ over the next time horizon and tracking the step-wise desired sequence of states $\mathcal{N}(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t)$ with a linear quadratic tracking controller.

from the forward variable with,

$$\boldsymbol{z}_t = \{z_t, \ldots, z_{T_p}\} = \arg\max_i \ \alpha_{t,i}^{\text{HSMM}}, \quad \hat{\boldsymbol{\mu}}_t = \tilde{\boldsymbol{\mu}}_{z_t}, \quad \hat{\boldsymbol{\Sigma}}_t = \tilde{\boldsymbol{\Sigma}}_{z_t}. \tag{3.42}$$

Fig. 3.4 shows a conceptual representation of the step-wise sequence of states generated by deterministically sampling from HSMM encoding of the Z-shaped data. In the next section, we show how to synthesise robot movement from this step-wise sequence of states in a smooth manner.

3.3 Decoding with Linear Quadratic Regulator/Tracking (LQR/LQT)

In this section, we combine the generative models with a commonly used tool from control theory to derive a control policy for the robot to perform manipulation tasks. The basic idea is to formulate the regulation of the desired pose $\mathcal{N}(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t)$ at time t or the tracking of the step-wise desired sequence of poses $\{\mathcal{N}(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t)\}_{t=1}^{T_p}$ as long-term optimization of a scalar cost function with a linear quadratic regulator (LQR) or a linear quadratic tracker

(LQT) respectively. Note that other alternatives such as trajectory-HSMM can also be used to smoothly follow the step-wise desired sequence of states [Zen 2007, Sugiura 2011]. We describe our formulation with the finite horizon case for tracking problem and then show how the infinite horizon case follows naturally from the finite horizon case for both continuous and discrete time linear systems.

3.3.1 Continuous LQR/LQT

The control policy $\boldsymbol{u}_t$ at each time step is obtained by minimizing the cost function over the **finite time horizon** T_p,

$$c_t(\boldsymbol{\xi}_t, \boldsymbol{u}_t) = \sum_{t=1}^{T_p} (\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_t)^\top \boldsymbol{Q}_t (\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_t) + \boldsymbol{u}_t^\top \boldsymbol{R}_t \boldsymbol{u}_t, \quad \text{s.t.} \quad \dot{\boldsymbol{\xi}}_t = \boldsymbol{A}_d \boldsymbol{\xi}_t + \boldsymbol{B}_d \boldsymbol{u}_t, \tag{3.43}$$

starting from the initial state $\boldsymbol{\xi}_1$ and following the linear dynamical system specified by $\boldsymbol{A}_d$ and $\boldsymbol{B}_d$. Without loss of generality, we consider a linear time-invariant double integrator system to describe the system dynamics. Alternatively, a time-varying linearization of the system dynamics along the reference trajectory can also be used to model the system dynamics as shown in the previous chapter. A physical analogue of the double integrator system is a unit mass attached to the datapoint $\boldsymbol{\xi}_t$ and the control input $\boldsymbol{u}_t$ applies a force to drive the unit mass with no friction. The double integrator is defined as

$$\overbrace{\begin{bmatrix} \dot{\boldsymbol{x}}_t \\ \ddot{\boldsymbol{x}}_t \end{bmatrix}}^{\dot{\boldsymbol{\xi}}_t} = \overbrace{\begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}}^{\boldsymbol{A}_d} \overbrace{\begin{bmatrix} \boldsymbol{x}_t \\ \dot{\boldsymbol{x}}_t \end{bmatrix}}^{\boldsymbol{\xi}_t} + \overbrace{\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix}}^{\boldsymbol{B}_d} \boldsymbol{u}_t, \tag{3.44}$$

with $\boldsymbol{\xi}_t = [\boldsymbol{x}_t{}^\top \ \dot{\boldsymbol{x}}_t{}^\top]^\top, \hat{\boldsymbol{\mu}}_t = [\hat{\boldsymbol{\mu}}_t^{x\top} \ \hat{\boldsymbol{\mu}}_t^{\dot{x}\top}]^\top$, $\boldsymbol{x}, \dot{\boldsymbol{x}}$ represent the position and velocity of the double integrator system, and $\hat{\boldsymbol{\mu}}_t^x, \hat{\boldsymbol{\mu}}_t^{\dot{x}}$ denote the desired position and velocity to follow. Setting $\boldsymbol{Q}_t = \hat{\boldsymbol{\Sigma}}_t^{-1} \succeq 0, \boldsymbol{R}_t \succ 0$, the control input $\boldsymbol{u}_t^*$ that minimizes the cost function is obtained by minimizing the Hamilton-Jacobi-Bellman equation [Bertsekas 2012],

$$\begin{aligned} \boldsymbol{u}_t^* &= -\boldsymbol{R}_t^{-1} \boldsymbol{B}_d^\top \boldsymbol{P}_t (\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_t) + \boldsymbol{R}_t^{-1} \boldsymbol{B}_d^\top \boldsymbol{d}_t, \\ &= \boldsymbol{K}_t^{\mathcal{P}} (\hat{\boldsymbol{\mu}}_t^x - \boldsymbol{x}_t) + \boldsymbol{K}_t^{\mathcal{V}} (\hat{\boldsymbol{\mu}}_t^{\dot{x}} - \dot{\boldsymbol{x}}_t) + \boldsymbol{R}_t^{-1} \boldsymbol{B}_d^\top \boldsymbol{d}_t, \end{aligned} \tag{3.45}$$

where $[\boldsymbol{K}_t^{\mathcal{P}}, \boldsymbol{K}_t^{\mathcal{V}}] = \boldsymbol{R}_t^{-1}\boldsymbol{B}_d^\top\boldsymbol{P}_t$ are the full stiffness and damping matrices, $\boldsymbol{R}_t^{-1}\boldsymbol{B}_d^\top\boldsymbol{d}_t$ is the feedforward term, and $\boldsymbol{P}_t, \boldsymbol{d}_t$ are the solutions of the following differential equations

$$\begin{aligned} -\dot{\boldsymbol{P}}_t &= \boldsymbol{A}_d^\top\boldsymbol{P}_t + \boldsymbol{P}_t\boldsymbol{A}_d - \boldsymbol{P}_t\boldsymbol{B}_d\boldsymbol{R}_t^{-1}\boldsymbol{B}_d^\top\boldsymbol{P}_t + \boldsymbol{Q}_t, \\ -\dot{\boldsymbol{d}}_t &= \boldsymbol{A}_d^\top\boldsymbol{d}_t - \boldsymbol{P}_t\boldsymbol{B}_d\boldsymbol{R}_t^{-1}\boldsymbol{B}_d^\top\boldsymbol{d}_t + \boldsymbol{P}_t\hat{\boldsymbol{\mu}}_t - \boldsymbol{P}_t\boldsymbol{A}_d\hat{\boldsymbol{\mu}}_t, \end{aligned} \tag{3.46}$$

with terminal conditions set to $\boldsymbol{P}_{T_p} = \boldsymbol{0}$ and $\boldsymbol{d}_{T_p} = \boldsymbol{0}$. Note that the gains can be precomputed before simulating the system if the reference trajectory does not change during the reproduction of the task. The resulting trajectory $\boldsymbol{\xi}_t^*$ smoothly tracks the step-wise reference trajectory $\hat{\boldsymbol{\mu}}_t$ and the gains $\boldsymbol{K}_t^{\mathcal{P}}, \boldsymbol{K}_t^{\mathcal{V}}$ stabilize $\boldsymbol{\xi}_t$ along $\boldsymbol{\xi}_t^*$ in accordance with the precision required during the task.

For the case of **infinite horizon** with $T_p \to \infty$ and $\boldsymbol{Q}_t = \boldsymbol{Q}$ in Eq. (3.3.1), the feedforward term is set to zero and $\boldsymbol{P}_{t-1} = \boldsymbol{P}_t = \boldsymbol{P}$ is obtained by minimizing the Continuous Algebraic Riccati Equation (CARE)

$$\boldsymbol{A}_d^\top\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A}_d - \boldsymbol{P}\boldsymbol{B}_d\boldsymbol{R}^{-1}\boldsymbol{B}_d^\top\boldsymbol{P} + \boldsymbol{Q} = \boldsymbol{0}. \tag{3.47}$$

To solve CARE, we define the Hamiltonian matrix

$$\boldsymbol{H}_a = \begin{bmatrix} \boldsymbol{A}_d & -\boldsymbol{B_d}\boldsymbol{R^{-1}}\boldsymbol{B_d^\top} \\ -\boldsymbol{Q} & -\boldsymbol{A_d^\top} \end{bmatrix}. \tag{3.48}$$

Eigendecomposition of the Hamiltonian matrix is used to extract the subspace of $\boldsymbol{H}_a$ with negative real eigenvalues defined as, $\begin{bmatrix} \boldsymbol{V}_1 \\ \boldsymbol{V}_{21} \end{bmatrix}$, i.e.,

$$\boldsymbol{H}_a = \boldsymbol{V}\begin{bmatrix} \boldsymbol{\lambda}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\lambda}_2 \end{bmatrix}\boldsymbol{V}^\top, \quad \text{with} \quad \boldsymbol{V} = \begin{bmatrix} \boldsymbol{V}_1 & \boldsymbol{V}_{12} \\ \boldsymbol{V}_{21} & \boldsymbol{V}_2 \end{bmatrix}. \tag{3.49}$$

The solution of algebraic Riccati equation solution gives $\boldsymbol{P} = \boldsymbol{V}_{21}\boldsymbol{V}_1^{-1}$. The control law for the infinite horizon case can now be expressed as

$$\boldsymbol{u}_t^* = -\boldsymbol{R}^{-1}\boldsymbol{B}_d^\top\boldsymbol{P}(\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_{\boldsymbol{t}}). \tag{3.50}$$

The value function for the infinite horizon case, $V(\boldsymbol{\xi}_t) = (\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_{\boldsymbol{t}})^\top\boldsymbol{P}(\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_{\boldsymbol{t}})$, reveals that the control law moves in the steepest descent direction of the value function $-\boldsymbol{P}(\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_{\boldsymbol{t}})$. The descent direction is projected onto the control space with $\boldsymbol{B}_d$ and scaled with different weights using $\boldsymbol{R}^{-1}$.

3.3.2 Discrete LQR/LQT

The discrete-time dynamical system for the double integrator is defined as,

$$\overbrace{\begin{bmatrix} \boldsymbol{x}_{t+1} \\ \boldsymbol{x}_{t+2} \end{bmatrix}}^{\boldsymbol{\xi}_{t+1}} = \overbrace{\begin{bmatrix} \boldsymbol{I} & \boldsymbol{\Delta} t \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}}^{\boldsymbol{A}_d} \overbrace{\begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{x}_{t+1} \end{bmatrix}}^{\boldsymbol{\xi}_t} + \overbrace{\begin{bmatrix} \boldsymbol{I}\frac{1}{2}\Delta t^2 \\ \boldsymbol{I}\Delta t \end{bmatrix}}^{\boldsymbol{B}_d} \boldsymbol{u}_t. \tag{3.51}$$

The control law $\boldsymbol{u}_t^*$ that minimizes the cost function in Eq. (3.3.1) under **finite horizon** subject to the linear dynamics in discrete time is given as,

$$\begin{aligned} \boldsymbol{u}_t^* &= -\left(\boldsymbol{R} + \boldsymbol{B}_d^\top \boldsymbol{P}_t \boldsymbol{B}_d\right)^{-1} \boldsymbol{B}_d^\top \boldsymbol{P}_t \boldsymbol{A}_d \left(\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_t\right) - \left(\boldsymbol{R} + \boldsymbol{B}_d^\top \boldsymbol{P}_t \boldsymbol{B}_d\right)^{-1} \boldsymbol{B}_d^\top \left(\boldsymbol{P}_t \left(\boldsymbol{A}_d \hat{\boldsymbol{\mu}}_t - \hat{\boldsymbol{\mu}}_t\right) + \boldsymbol{d}_t\right), \\ &= \boldsymbol{K}_t^{\mathcal{P}} (\hat{\boldsymbol{\mu}}_t^x - \boldsymbol{x}_t) + \boldsymbol{K}_t^{\mathcal{V}} (\hat{\boldsymbol{\mu}}_t^{\dot{x}} - \dot{\boldsymbol{x}}_t) - \left(\boldsymbol{R} + \boldsymbol{B}_d^\top \boldsymbol{P}_t \boldsymbol{B}_d\right)^{-1} \boldsymbol{B}_d^\top \left(\boldsymbol{P}_t \left(\boldsymbol{A}_d \hat{\boldsymbol{\mu}}_t - \hat{\boldsymbol{\mu}}_t\right) + \boldsymbol{d}_t\right), \end{aligned} \tag{3.52}$$

where $[\boldsymbol{K}_t^{\mathcal{P}}, \boldsymbol{K}_t^{\mathcal{V}}] = -\left(\boldsymbol{R} + \boldsymbol{B}_d^\top \boldsymbol{P}_t \boldsymbol{B}_d\right)^{-1} \boldsymbol{B}_d^\top \boldsymbol{P}_t \boldsymbol{A}_d$ are the full stiffness and damping matrices for the feedback term, and $\left(\boldsymbol{R} + \boldsymbol{B}_d^\top \boldsymbol{P}_t \boldsymbol{B}_d\right)^{-1} \boldsymbol{B}_d^\top \left(\boldsymbol{P}_t \left(\boldsymbol{A}_d \hat{\boldsymbol{\mu}}_t - \hat{\boldsymbol{\mu}}_t\right) + \boldsymbol{d}_t\right)$ is the feedforward term. $\boldsymbol{P}_t$ and $\boldsymbol{d}_t$ are respectively obtained by solving the Riccati differential equation and linear differential equation backwards in discrete time from terminal conditions $\boldsymbol{P}_{T_p} = \boldsymbol{Q}_{T_p}$ and $\boldsymbol{d}_{T_p} = \boldsymbol{0}$,

$$\boldsymbol{P}_{t-1} = \boldsymbol{Q}_t - \boldsymbol{A}_d^\top \Big(\boldsymbol{P}_t \boldsymbol{B}_d \left(\boldsymbol{R} + \boldsymbol{B}_d^\top \boldsymbol{P}_t \boldsymbol{B}_d\right)^{-1} \boldsymbol{B}_d^\top \boldsymbol{P}_t - \boldsymbol{P}_t\Big) \boldsymbol{A}_d, \tag{3.53}$$

$$\boldsymbol{d}_{t-1} = \Big(\boldsymbol{A}_d^\top - \boldsymbol{A}_d^\top \boldsymbol{P}_t \boldsymbol{B}_d \left(\boldsymbol{R} + \boldsymbol{B}_d^\top \boldsymbol{P}_t \boldsymbol{B}_d\right)^{-1} \boldsymbol{B}_d^\top\Big) \Big(\boldsymbol{P}_t \left(\boldsymbol{A}_d \hat{\boldsymbol{\mu}}_t - \hat{\boldsymbol{\mu}}_{t+1}\right) + \boldsymbol{d}_t\Big). \tag{3.54}$$

For the **infinite horizon** case with $T \to \infty$ and the desired pose $\hat{\boldsymbol{\mu}}_t = \hat{\boldsymbol{\mu}}_{t_0}$, the control law in (3.52) remains the same except the feedforward term is set to zero and $\boldsymbol{P}_{t-1} = \boldsymbol{P}_t = \boldsymbol{P}$ is the steady-state solution obtained by eigen value decomposition of the discrete algebraic Riccati equation (DARE) in (3.53) [Borrelli 2011]. To solve DARE, we define the symplectic matrix,

$$\boldsymbol{H}_b = \begin{bmatrix} \boldsymbol{A}_d + \boldsymbol{B}_d \boldsymbol{R}^{-1} \boldsymbol{B}_d^\top (\boldsymbol{A}_d^{-1})^\top \boldsymbol{Q} & \boldsymbol{B}_d \boldsymbol{R}^{-1} \boldsymbol{B}_d^\top (\boldsymbol{A}_d^{-1})^\top \\ -(\boldsymbol{A}_d^{-1})^\top \boldsymbol{Q} & (\boldsymbol{A}_d^{-1})^\top \end{bmatrix}. \tag{3.55}$$

The eigenvectors of $\boldsymbol{H}_b$ corresponding to eigenvalues lying inside the unit circle are used to solve DARE. Let $\begin{bmatrix} \boldsymbol{V}_1^\top & \boldsymbol{V}_{21}^\top \end{bmatrix}^\top$ denote the corresponding subspace of $\boldsymbol{H}_b$, then the solution of DARE is, $\boldsymbol{P} = \boldsymbol{V}_{21} \boldsymbol{V}_1^{-1}$ and the control law takes the form,

$$\boldsymbol{u}_t^* = -(\boldsymbol{R} + \boldsymbol{B}_d^\top \boldsymbol{P} \boldsymbol{B}_d)^{-1} \boldsymbol{B}_d^\top \boldsymbol{P} \boldsymbol{A}_d (\boldsymbol{\xi}_t - \hat{\boldsymbol{\mu}}_{\boldsymbol{t}}). \tag{3.56}$$

Both discrete and continuous time linear quadratic regulator/tracker can be used to follow

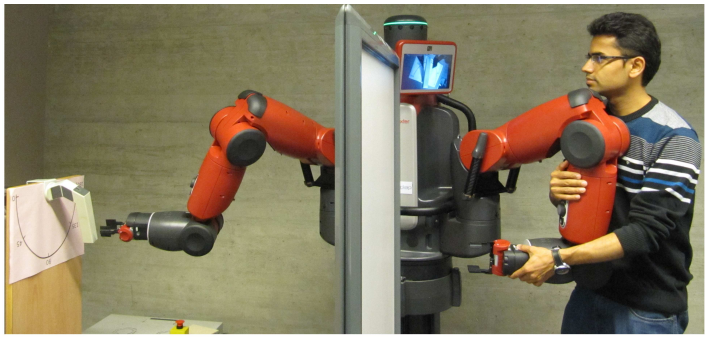


Figure 3.5: Baxter robot learns to open/close the valve from previously unseen configurations.

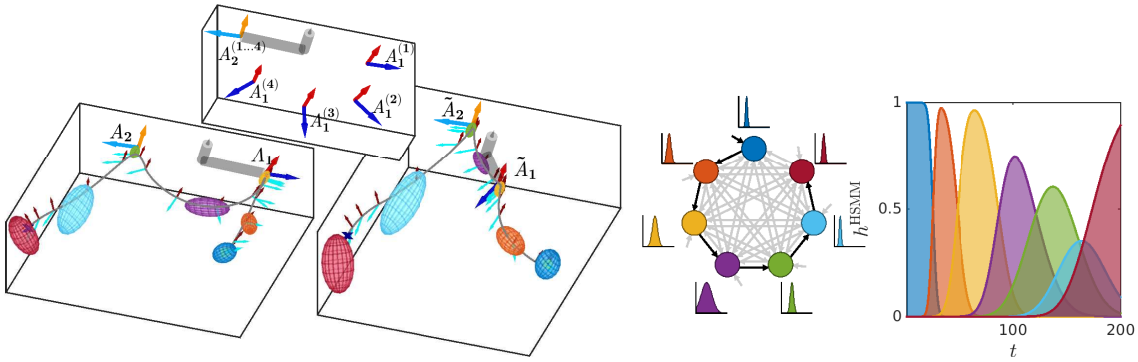


Figure 3.6: Baxter valve opening task: *(left)* valve opening movement reproduction for a training set on left and for an unseen valve configuration on right, *(right)* resulting left-right HSMM encoding of the task with duration model shown next to each state ($s^{\max} = 100$) on left, and rescaled forward variable, $h_{t,i}^{\text{HSMM}} = \frac{\alpha_{t,i}^{\text{HSMM}}}{\sum_{k=1}^{K} \alpha_{t,k}^{\text{HSMM}}}$, evolution with time on right.

the desired pose/trajectory. The discrete time formulation, however, gives numerically stable results for a wide range of values of $\boldsymbol{R}$. A similar treatment of decoding HSMM with a batch solution of control inputs can be found in [Zeestraten 2016]. Fig. 3.4 shows the results of applying discrete LQT on the desired step-wise sequence of states sampled from an HSMM encoding the Z-shaped demonstrations.

3.4 Valve Opening Example

Valve opening task is a standard benchmark in robotics because it can be applied to a wide range of environments and applications. The goal is to bring the valve in an open position from different initial configurations of the valve using the torque-controlled Baxter robot as shown in Fig. 3.5.

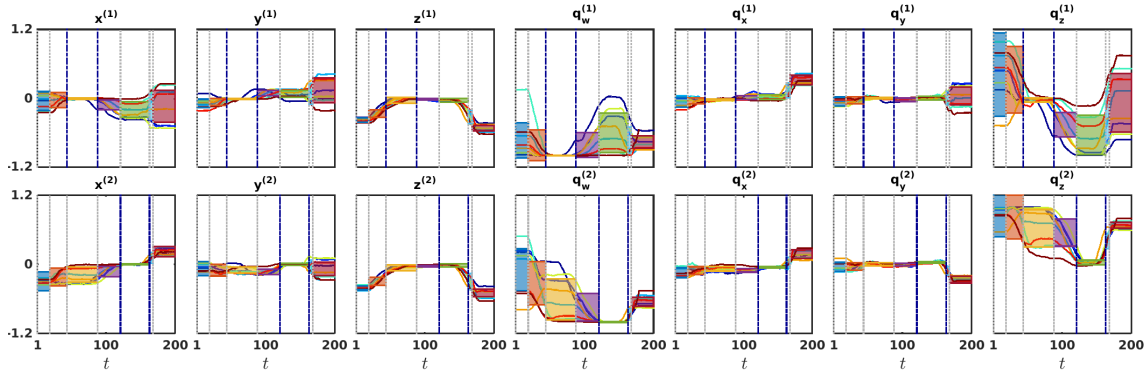


Figure 3.7: Variance of the learned model along position and orientation variables in coordinate system 1 *(top)* and coordinate system 2 *(bottom)*. Invariant phase across all demonstrations (highlighted in blue) is observed for components 3 and 5 in frame 1 and frame 2 respectively.

The adaptive aspect of the task requires to ascertain where to grasp the valve and where to stop turning it. Consequently, we attach two frames, one with the observed initial configuration of valve $\{\boldsymbol{A}_1, \boldsymbol{b}_1\}$ and other with the desired end configuration of the valve $\{\boldsymbol{A}_2, \boldsymbol{b}_2\}$ (marked with a visual tag of 0 degree around the valve). We record eight kinesthetic demonstrations with the initial configuration of the valve corresponding to $\{180, 135, 90, 45, 157.5, 112.5, 67.5, 22.5\}$ degrees with the horizontal in the successive demonstrations, $n = 1 \ldots 8$. The first 4 demonstrations are used for the training test, while the remaining 4 are used for the test set. Each observation comprises of the end-effector Cartesian position $\boldsymbol{x}_t^p \in \mathbb{R}^3$, quaternion orientation $\boldsymbol{\varepsilon}_t^o \in \mathbb{R}^4$, linear velocity $\dot{\boldsymbol{x}}_t^p \in \mathbb{R}^3$, and quaternion derivative (estimated from angular velocity) $\dot{\boldsymbol{\varepsilon}}_t^o \in \mathbb{R}^4$ for a total of 14 dimensions per sample. Each demonstration is further downsampled to a total of 200 datapoints. For notational convenience, we define $\boldsymbol{\xi}_t = [\boldsymbol{x}_t{}^\top \ \dot{\boldsymbol{x}}_t{}^\top]^\top$ with $\boldsymbol{x}_t = \left[\boldsymbol{x}_t^{p\top} \ \boldsymbol{\varepsilon}_t^{o\top}\right]^\top$ and $\dot{\boldsymbol{x}}_t = \left[\dot{\boldsymbol{x}}_t^{p\top} \ \dot{\boldsymbol{\varepsilon}}_t^{o\top}\right]^\top$, and represent the frame as

$$\boldsymbol{A}_j^{(n)} = \begin{bmatrix} \boldsymbol{R}_j^{(n)} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \mathcal{E}_j^{(n)} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{R}_j^{(n)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \mathcal{E}_j^{(n)} \end{bmatrix}, \boldsymbol{b}_j^{(n)} = \begin{bmatrix} \boldsymbol{p}_j^{(n)} \\ \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \tag{3.57}$$

where $\boldsymbol{p}_j^{(n)} \in \mathbb{R}^3, \boldsymbol{R}_j^{(n)} \in \mathbb{R}^{3\times 3}, \mathcal{E}_j^{(n)} \in \mathbb{R}^{4\times 4}$ denote the Cartesian position, the rotation matrix and the quaternion matrix of the j-th frame in the n-th demonstration respectively. A sketch of different frames in the demonstrations can be seen in top zoomed portion of Fig. 3.6. Note that we do not consider time as an explicit variable as the duration model in HSMM encapsulates the timing information locally.

The number of Gaussians are empirically selected in this experiment based on the important phases in the task such as reaching, grasping, turning etc. Alternatively, a Bayesian information criterion, or a non-parametric approach based on Dirichlet processes can also be used for model selection as we will see in Chap. 5. Performance setting in our experiments is as follows: $\{\pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{K}$ are initialized using k-means clustering algorithm, $\boldsymbol{B} = 0.1\boldsymbol{I}, \boldsymbol{R} = 9\boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix. Results of regenerating the movements with 7 mixture components are shown in Fig. 3.6. For a given initial configuration of the valve, the model parameters are adapted by evaluating the product of Gaussians for a new frame configuration. The reference trajectory is then computed from the initial position of the robot arm using the forward variable (see Fig. 3.6 for HSMM encoding) and tracked using LQT. The robot arm moves from its initial configuration to align itself with the first frame $\{\boldsymbol{A}_1, \boldsymbol{b}_1\}$ to grasp the valve, and follows it with the turning movement to align with the second frame $\{\boldsymbol{A}_2, \boldsymbol{b}_2\}$ before returning back to the home position. Fig. 3.7 shows that the task-parameterized formulation exploits variability in the observed demonstrations to statistically encode different phases of the task. Here, reaching the valve and coming back to home position have higher variability in the demonstrations, whereas aligning with the frames for grasping/turning and stopping the valve have no observed variations in their respective coordinate systems. Consequently, the robot arm is able to reach the valve from different initial configurations, grasp the valve and turn it to the desired position.

3.5 Conclusion

In this chapter, we have proposed a framework combining generative models, task adaptability and optimal control for learning and reproduction of robot manipulation tasks. The hidden semi-Markov model approximates the probability density function of the demonstrations and segments the demonstrations into meaningful components. Task-parameterization of the model enables the robot to readily adopt for better generalization in previously unseen environmental situations. By sampling the sequence of states from the model and following them with a linear quadratic tracking controller, we are able to autonomously perform manipulation tasks in a smooth manner. An interested reader is encouraged to see [Pignat 2017] for other robotic applications derived from the framework presented here for learning robot manipulation skills.

Chapter 4

Scalable Generative Models in Latent Space

Contents

4.1 Subspace Clustering **80**
4.1.1 High-Dimensional Data Clustering 81
4.1.2 Mixture of Factor Analyzers (MFA) Decomposition 82
4.2 Semi-Tied Mixture Model **84**
4.2.1 Maximum Likelihood Parameter Estimation 86
4.2.2 Analysis of Semi-Tied Mixture Models 88
4.2.3 Whole Body Motion Capture Data - Chicken Dance Example 88
4.3 Task-Parameterized HSMM in Latent Space **89**
4.3.1 Valve Opening Comparison 92
4.3.2 Pick-and-Place with Obstacle Avoidance Example 92
4.4 Conclusion **94**

Dimensionality reduction has long been recognized as a fundamental problem in unsupervised learning. Classical model-based generative models tend to suffer from the *curse of dimensionality* when few datapoints are available, as in the case of robot learning from demonstrations. Statistical subspace clustering methods address this challenge by using a parsimonious model to reduce the number of parameters that can be robustly estimated. A simple way to reduce the number of parameters would be to constrain the covariance structure to a diagonal or spherical/isotropic matrix, thereby, restricting the number of parameters at the cost of treating each dimension separately. Such decoupling, however,

cannot encode the important motor control principles of coordination, synergies and action-perception couplings [Wolpert 2011].

In this chapter, we seek out a latent feature space in the high-dimensional data to reduce the number of model parameters that can be robustly estimated. The role of latent space is to decorrelate the data so that the mixture components map the data onto the corresponding subspaces to cope with insufficient or noisy training data. We base our formulation on low-rank decomposition of the covariance matrix using *Mixture of Factor Analyzers* (MFA) approach [McLachlan 2003]. We then exploit a technique for partially tying the covariance matrices of the mixture model [Gales 1999]. The technique associates or ties the covariance matrices of the mixture model with a common latent space, and only uses a diagonal matrix for appropriate scaling of the basis vectors in the latent space. We combine these latent space models with hidden semi-Markov model and linear quadratic tracking controller for encapsulating reactive autonomous behaviour as shown in the previous chapter, and show the suitability of our approach for learning manipulation tasks in robotics with the task-parameterized formulation [Tanwani 2016a].

4.1 Subspace Clustering

Gaussian mixture models approximate the probability density function of the demonstrations as a convex combination of K multivariate Gaussian distributions, each having a mixing coefficient π_i, mean $\boldsymbol{\mu}_i$, and a covariance matrix $\boldsymbol{\Sigma}_i$. The number of parameters in the covariance matrix $\boldsymbol{\Sigma}_i$ grows quadratically with the dimension of datapoints D, leading to poor performance in high-dimensional spaces. Some typical solutions employed in practice to address this challenge include [Bouveyron 2014]

- **Regularization:** To avoid numerical problems in inverting covariance matrices, a simple regularization term σ is added to each covariance matrix $\boldsymbol{\Sigma}_i \leftarrow \boldsymbol{\Sigma}_i + \sigma \boldsymbol{I}$ during the maximization step of the EM loop. Other important regularization forms include lasso and ridge regression in estimating covariance matrix.

- **Dimensionality Reduction:** Most of the work on clustering models in high dimensional spaces has focused on global dimensionality reduction methods as a pre-processing step. Notable examples include principal component analysis (PCA), factor analysis (FA) and linear discriminant analysis (LDA).

- **Parsimonious models:** To avoid over-fitting, a parsimonious structure is imposed on the covariance matrix with fewer model parameters. Common exam-

ples in this category include isotropic/spherical covariance, diagonal covariance, and block-diagonal covariance. Note that the diagonal covariance structure corresponds to a separate treatment of each variable. The assumption is also made in movement encoding with DMPs [Ijspeert 2013] that are widely used in many robotics applications. Such assumptions, however, discards the important synergistic information among the variables shown in several motor control studies [Mussa-Ivaldi 1994, Todorov 2002, Hogan 2012].

There are alternatives in learning mixture models in between the diagonal and the full covariances that have rarely been explored in the context of robot skills acquisition. These alternatives can be studied as a subspace clustering problem, that aims at grouping the data such that they can be locally projected in a subspace of reduced dimensionality. Subspace clustering models learn multiple subspaces to encode the data according to their local trend, i.e., they perform segmentation and dimensionality reduction simultaneously. Note that subspace clustering is not the same as performing clustering and dimensionality reduction separately [Ghahramani 1997]. A broad range of these models encompasses sparse subspace clustering [Elhamifar 2013], DP-space clustering [Wang 2015], high-dimensional data clustering (HDDC) [Bouveyron 2007], parsimonious models [Bouveyron 2014], mixture of factor analyzers (MFA) [McLachlan 2003] or mixture of probabilistic principal component analyzers (MPPCA) [Tipping 1999]. We briefly review here a couple of pertinent methods for statistical subspace clustering (see also [Calinon 2016] for a review).

4.1.1 High-Dimensional Data Clustering

High-dimensional data clustering (HDDC) performs both subspace clustering and regularization by modeling each cluster with a set of d_i principal eigenvectors ($d_i < D$) corresponding to eigenvalues λ_{ij}, and uses a spherical variance for the remaining directions with the eigenvalue $\bar{\lambda}_i$ such that

$$\bar{\lambda}_i = \frac{1}{D - d_i} \sum_{j=d_i+1}^{D} \lambda_{ij}, \tag{4.1}$$

The eigenvalue $\bar{\lambda}_i$ replaces the last $D - d_i$ eigenvalues of $\mathbf{\Sigma}_i$ in order to reconstruct a full covariance matrix (see also [Bouveyron 2007]).

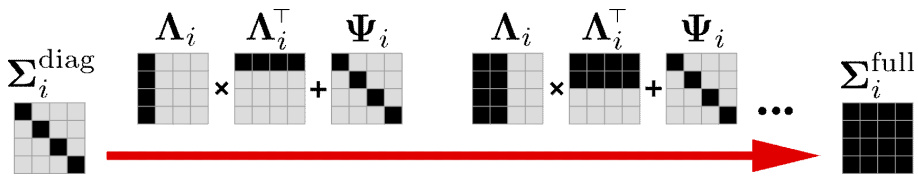


Figure 4.1: Parameters representation of a diagonal, full and mixture of factor analyzers decomposition of covariance matrix. Filled blocks represent non-zero entries.

4.1.2 Mixture of Factor Analyzers (MFA) Decomposition

The basic idea of MFA is to perform subspace clustering by assuming the covariance structure for each component of the form,

$$\boldsymbol{\Sigma}_i = \boldsymbol{\Lambda}_i \boldsymbol{\Lambda}_i^\top + \boldsymbol{\Psi}_i, \tag{4.2}$$

where $\boldsymbol{\Lambda_i} \in \mathbb{R}^{D\times d}$ is the *factor loadings matrix* with $d < D$ for parsimonious representation of the data, and $\boldsymbol{\Psi}_i$ is the diagonal noise matrix (see Fig. 4.1 for MFA representation in comparison to a diagonal and a full covariance matrix). Further structure can be imposed on the factor loading matrix and the noise matrix to yield a family of parsimonious models [McNicholas 2008]. For example, the mixture of probabilistic principal component analysis (MPPCA) model is a special case of MFA with the distribution of the errors assumed to be isotropic with $\boldsymbol{\Psi}_i = \boldsymbol{I}\sigma_i^2$ [Tipping 1999]. The MFA model assumes that $\boldsymbol{\xi}_t$ is generated using a linear transformation of d-dimensional vector of latent (unobserved) factors $\boldsymbol{f}_t$,

$$\boldsymbol{\xi}_t = \boldsymbol{\Lambda}_i \boldsymbol{f}_t + \boldsymbol{\mu}_i + \boldsymbol{\varepsilon}, \tag{4.3}$$

where $\boldsymbol{\mu}_i \in \mathbb{R}^D$ is the mean vector of the i-th factor analyzer, $\boldsymbol{f}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is a normally distributed factor, and $\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Psi}_i)$ is a zero-mean Gaussian noise with diagonal covariance $\boldsymbol{\Psi}_i$. The diagonal assumption implies that the observed variables are independent given the factors. The goal of MFA is to model the covariance structure of $\boldsymbol{\xi}_t$ such that,

$$\boldsymbol{\xi}_t \sim \mathcal{N}\left(\boldsymbol{\mu}_i,\ \boldsymbol{\Lambda}_i \boldsymbol{\Lambda}_i^\top + \boldsymbol{\Psi}_i\right), \tag{4.4}$$

where the joint distribution of $\boldsymbol{\xi}_t$ and $\boldsymbol{f}_t$ is,

$$\begin{bmatrix} \boldsymbol{\xi}_t \\ \boldsymbol{f}_t \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_i \\ \boldsymbol{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Lambda}_i \boldsymbol{\Lambda}_i^\top + \boldsymbol{\Psi}_i & \boldsymbol{\Lambda}_i \\ \boldsymbol{\Lambda}_i^\top & \boldsymbol{I} \end{bmatrix} \right). \tag{4.5}$$

The model parameters $\theta = \{\pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Lambda}_i, \boldsymbol{\Psi}_i\}_{i=1}^K$ are estimated from the data using an EM algorithm [Ghahramani 1997, McNicholas 2008] summarized as

E-step:

$$h_{t,i} = \frac{\pi_i \, \mathcal{N}(\boldsymbol{\xi}_t \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{k=1}^{K} \pi_k \, \mathcal{N}(\boldsymbol{\xi}_t \mid \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_k^\top + \boldsymbol{\Psi}_k)}. \tag{4.6}$$

M-step:

$$\pi_i \leftarrow \frac{\sum_{t=1}^{T} h_{t,i}}{T}, \tag{4.7}$$

$$\boldsymbol{\mu}_i \leftarrow \frac{\sum_{t=1}^{T} h_{t,i} \boldsymbol{\xi}_t}{\sum_{t=1}^{T} h_{t,i}}, \tag{4.8}$$

$$\boldsymbol{\Lambda}_i \leftarrow \boldsymbol{S}_i \boldsymbol{B}_i^\top (\boldsymbol{I} - \boldsymbol{B}_i \boldsymbol{\Lambda}_i + \boldsymbol{B}_i \boldsymbol{S}_i \boldsymbol{B}_i^\top)^{-1}, \tag{4.9}$$

$$\boldsymbol{\Psi}_i \leftarrow \operatorname{diag}\left(\operatorname{diag}\left(\boldsymbol{S}_i - \boldsymbol{\Lambda}_i \boldsymbol{B}_i \boldsymbol{S}_i\right)\right), \tag{4.10}$$

$$\boldsymbol{\Sigma}_i \leftarrow \boldsymbol{\Lambda}_i \boldsymbol{\Lambda}_i^\top + \boldsymbol{\Psi}_i, \tag{4.11}$$

where $\boldsymbol{S}_i$ is the sample covariance matrix and $\boldsymbol{B}_i$ is the projection of $\boldsymbol{\xi}_t$ to the latent space such that, $\boldsymbol{z}|\boldsymbol{\xi}_t \sim \boldsymbol{B}_i \, (\boldsymbol{\mu}_i - \boldsymbol{\xi}_t)$ with,

$$\boldsymbol{S}_i = \frac{\sum_{t=1}^{T} h_{t,i} (\boldsymbol{\xi}_t - \boldsymbol{\mu}_i)(\boldsymbol{\xi}_t - \boldsymbol{\mu}_i)^\top}{\sum_{t=1}^{T} h_{t,i}}, \tag{4.12}$$

$$\boldsymbol{B}_i = \boldsymbol{\Lambda}_i^\top (\boldsymbol{\Lambda}_i \boldsymbol{\Lambda}_i^\top + \boldsymbol{\Psi}_i)^{-1}. \tag{4.13}$$

For comparison, the M-step in MPPCA is given by [Tipping 1999],

$$\boldsymbol{\Lambda}_i \leftarrow \boldsymbol{S}_i \boldsymbol{\Lambda}_i (\boldsymbol{I} \sigma_i^2 + \boldsymbol{\Sigma}_i^{\top -1} \boldsymbol{\Lambda}_i^\top \boldsymbol{S}_i \boldsymbol{\Lambda}_i)^{-1}, \tag{4.14}$$

$$\sigma_i^2 \leftarrow \frac{1}{D} \operatorname{tr}(\boldsymbol{S}_i - \boldsymbol{S}_i \boldsymbol{\Lambda}_i \boldsymbol{\Sigma}_i^{\top -1} \boldsymbol{\Lambda}_i^\top), \tag{4.15}$$

$$\boldsymbol{\Sigma}_i \leftarrow \boldsymbol{\Lambda}_i \boldsymbol{\Lambda}_i^\top + \sigma_i^2 \boldsymbol{I}. \tag{4.16}$$

The MFA modeling approach can be combined with deep learning strategies to learn a hierarchical structure of layers in latent space [Tang 2012]. Coordinated MFA has found its application in robotics in tracking 3D human movement from motion capture data [R. Li 2010], and more recently for learning trajectories in robot programming by demonstration framework [Field 2015].

The hypothesis of MFA models can be viewed as less restrictive than HDDC models based on eigendecomposition since the subspace of each class does not need to be spanned by orthogonal vectors, whereas it is a necessary condition in models based on eigendecomposition such as PCA [Bouveyron 2007].

Note that each covariance matrix of the mixture component in HDDC and MPPCA has

its own subspace spanned by the basis vectors of $\boldsymbol{\Sigma}_i$. As the number of components increase to encode more complex skills, an increasing large number of potentially redundant parameters are used to fit the data. Consequently, we advocate the need to share the basis vectors across the mixture components. This concept was first exploited in speech processing where the covariance matrices in output state sequence of a Hidden Markov Model (HMM) were tied to a common linear transform [Gales 1999]. Parameter tying, for example, has been used to robustly estimate the density parameters with thousands of states in a HMM for building phone models [Leggetter 1995].

To the best of our knowledge, the concept of tying covariance matrices in mixture models to encode manipulation skills in robotics has not been used. We are interested in exploiting the coordination patterns in the demonstrations by semi-tying the model parameters, while reducing the number of parameters that can be robustly estimated. We extend the method to a task-parameterized model and encode the state duration and transition with a hidden semi-Markov model to enable the handling of previously unseen situations in an autonomous manner as seen in the previous chapter.

4.2 Semi-Tied Mixture Model

When the covariance matrices of the mixture model share the same set of parameters for the latent feature space, we call the model a *semi-tied* Gaussian mixture model. The main idea behind semi-tied GMMs is to decompose the covariance matrix $\boldsymbol{\Sigma}_i$ into two terms: a common latent feature matrix $\boldsymbol{H} \in \mathbb{R}^{D\times D}$ and a component-specific diagonal matrix $\boldsymbol{\Sigma}_i^{(\text{diag})} \in \mathbb{R}^{D\times D}$, i.e.,

$$\boldsymbol{\Sigma}_i = \boldsymbol{H}\boldsymbol{\Sigma}_i^{(\text{diag})}\boldsymbol{H}^\top. \tag{4.17}$$

The latent feature matrix encodes the locally important synergistic directions represented by D non-orthogonal basis vectors that are shared across all the mixture components, while the diagonal matrix selects the appropriate subspace of each mixture component as convex combination of a subset of the basis vectors of $\boldsymbol{H}$. Depending upon the sparsity of the convex combination, there are multiple subspaces to choose. In other words, we search for a global linear transformation of the data such that the transformed data can be modelled by a mixture of diagonal covariance matrices only.[1]

In high-dimensional spaces, Gaussian mixture components with full covariance matrices

[1]Note that the eigen decomposition of $\boldsymbol{\Sigma}_i = \boldsymbol{U}_i\boldsymbol{\Sigma}_i^{(\text{diag})}\boldsymbol{U}_i^\top$ contains D basis vectors of $\boldsymbol{\Sigma}_i$ in $\boldsymbol{U}_i$. In comparison, semi-tied mixture model gives D globally representative basis vectors that are shared across all the mixture components.

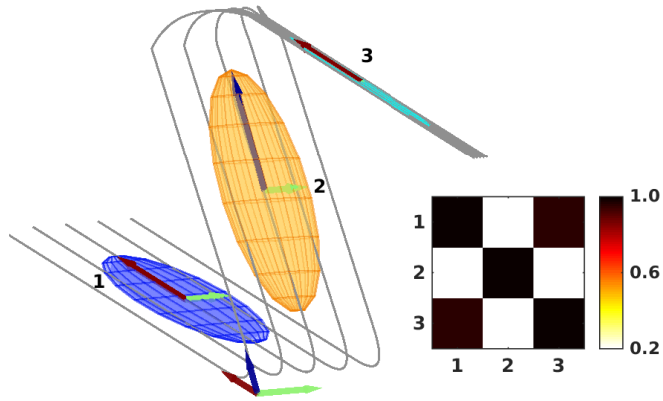


Figure 4.2: *(left)* Semi-tied mixture model encoding of Z-shaped data with 3 components and basis vectors shown at the origin, *(right)* pairwise correlation among the mixture components of semi-tied GMM (see section 4.2.2 for details).

tend to over-fit the training data when the data is noisy and/or the number of datapoints is insufficient. By tying the covariance matrices, the mixture components are forced to align along a set of common coordination patterns. This is also in line with biological motor control where the central nervous system (CNS) is believed to generate complex movements by temporal modulation of postural synergies [d'Avella 2003]. The implementation of postural synergies corresponds here to the basis vectors of $\boldsymbol{H}$, while the diagonal matrix of each mixture component $\boldsymbol{\Sigma}_i^{(\text{diag})}$ modulates the basis vectors in time for efficient encoding of complex tasks.

To illustrate the concept of semi-tied model parameters, consider the 3-dimensional Z-shaped demonstrations in Fig. 4.2. Encoding with semi-tied GMM reveals the locally important basis vectors comprising the latent feature space $\boldsymbol{H}$. In contrast, PCA here would yield orthogonal basis vectors along the directions of largest variance globally. Note that the basis vectors are not required to be orthogonal in the semi-tied GMM. It can be seen in Fig. 4.2 that the basis vector in red is shared across the first and the third mixture component, while the basis vector in green is shared across the first and the second mixture component. The basis vector in blue is tied only to the second mixture component. This yields high correlation between the first and the third mixture component, and low correlation of the second Gaussian component with other mixture components (see right of Fig. 4.2).

4.2.1 Maximum Likelihood Parameter Estimation

We are interested in maximum likelihood estimates of the parameters of semi-tied GMM, $\theta = \{\{\pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i^{(\text{diag})}\}_{i=1}^K, \boldsymbol{H}\}$. Given the initial set of parameters $\hat{\theta}$, substituting the expression for $\boldsymbol{\Sigma}_i$ from Eq. (4.17) in the auxiliary function [Dempster 1977] yields,

$$\begin{aligned}\mathcal{Q}(\theta, \hat{\theta}) &\approx \frac{1}{2}\sum_{t=1}^{T}\sum_{i=1}^{K} h_{t,i}^{\hat{\theta}} \left(\log\left(\pi_i^2 - |\boldsymbol{\Sigma}_i|\right) - \boldsymbol{\xi}_t^{i\top}\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\xi}_t^i\right), \\ &\approx \frac{1}{2}\sum_{t=1}^{T}\sum_{i=1}^{K} h_{t,i}^{\hat{\theta}} \left(2\log(\pi_i) - \log\left(\frac{|\boldsymbol{\Sigma}_i^{(\text{diag})}|}{|\boldsymbol{B}|^2}\right) - \boldsymbol{\xi}_t^{i\top}\boldsymbol{B}^\top\boldsymbol{\Sigma}_i^{(\text{diag})-1}\boldsymbol{B}\boldsymbol{\xi}_t^i\right), \end{aligned} \tag{4.18}$$

where $\boldsymbol{B} = \boldsymbol{H}^{-1}, \boldsymbol{\xi}_t^i = \boldsymbol{\xi}_t - \boldsymbol{\mu}_i$, and $h_{t,i}^{\hat{\theta}} = p(i|\boldsymbol{\xi}_t, \hat{\theta})$ is the probability of data point $\boldsymbol{\xi}_t$ to belong to i-th Gaussian component at time t. Setting $\frac{\partial\mathcal{Q}(\theta,\hat{\theta})}{\partial\boldsymbol{B}}$ and $\frac{\partial\mathcal{Q}(\theta,\hat{\theta})}{\partial\boldsymbol{\Sigma}_i^{(\text{diag})}}$ equal to 0, and solving for $\boldsymbol{B}$ and $\boldsymbol{\Sigma}_i^{(\text{diag})}$ respectively results in an expectation-maximization procedure to compute the maximum likelihood estimate of parameters (see [Gales 1998] for details). Following this, we get a row-by-row optimisation of $\boldsymbol{B}$, with $\boldsymbol{b}_d$ (d-th row of $\boldsymbol{B}$) related to all other rows by the cofactor of $\boldsymbol{B}$,

$$\boldsymbol{b}_d = \boldsymbol{c}_d\boldsymbol{G}_d^{-1}\sqrt{\frac{\sum_{t=1}^{T}\sum_{i=1}^{K} h_{t,i}^{\hat{\theta}}}{\boldsymbol{c}_d\boldsymbol{G}_d^{-1}\boldsymbol{c}_d^\top}}, \tag{4.19}$$

where $\boldsymbol{c}_d$ is the d-th row of cofactors of $\boldsymbol{B}$ with $\boldsymbol{C} = \text{cof}(\boldsymbol{B})$ recomputed after each update of $\boldsymbol{b}_d$,

$$\boldsymbol{C} = (\boldsymbol{B}^\top)^{-1}|\boldsymbol{B}|, \tag{4.20}$$

$$\boldsymbol{G}_d = \sum_{i=1}^{K}\frac{1}{\Sigma_{i,d}^{(\text{diag})}}\boldsymbol{S}_i\sum_{t=1}^{T} h_{t,i}^{\hat{\theta}}, \tag{4.21}$$

where $\Sigma_{i,d}^{(\text{diag})}$ is the d-th diagonal element of the i-th Gaussian, and $\boldsymbol{S}_i$ is the full sample covariance matrix given by

$$\boldsymbol{S}_i = \frac{\sum_{t=1}^{T} h_{t,i}^{\hat{\theta}}\,\boldsymbol{\xi}_t^i\,\boldsymbol{\xi}_t^{i\top}}{\sum_{t=1}^{T} h_{t,i}^{\hat{\theta}}}. \tag{4.22}$$

The corresponding maximum likelihood estimate of $\boldsymbol{\Sigma}_i^{(\text{diag})}$ is computed as

$$\boldsymbol{\Sigma}_i^{(\text{diag})} = \text{diag}\left(\boldsymbol{B}\boldsymbol{S}_i\boldsymbol{B}^\top\right). \tag{4.23}$$

Algorithm 4 *Semi-Tied Gaussian mixture model*

Input: $\{\{\boldsymbol{\xi}_t\}_{t=1}^T, K, \alpha^{\text{ST}}\}$
procedure EM Semi-Tied_GMM
Initialize the parameters: $\{\{\pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^K, \boldsymbol{B}\}$
repeat
  $h_{t,i}^{\hat{\theta}} := \frac{\pi_i \, \mathcal{N}(\boldsymbol{\xi}_t|\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i)}{\sum_{k=1}^K \pi_k \, \mathcal{N}(\boldsymbol{\xi}_t|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}$
  $\pi_i := \frac{\sum_{t=1}^T h_{t,i}^{\hat{\theta}}}{T}, \qquad \boldsymbol{\mu}_i := \frac{\sum_{t=1}^T h_{t,i}^{\hat{\theta}} \boldsymbol{\xi}_t}{\sum_{t=1}^T h_{t,i}^{\hat{\theta}}}$
  Compute $\boldsymbol{S}_i$ using Eq. (4.22)
  repeat
    Compute $\boldsymbol{\Sigma}_i^{(\text{diag})}$ using Eq. (4.23)
    for $d := 1$ **to** D **do**
      Compute $\boldsymbol{C}$ using Eq. (4.20)
      Compute $\boldsymbol{G}_d$ using Eq. (4.21)
      Compute $\boldsymbol{b}_d$ using Eq. (4.19)
    end for
  until $\boldsymbol{B}$ converges
  $\boldsymbol{H} := \boldsymbol{B}^{-1}$, compute $\boldsymbol{\Sigma}_i$ using Eq. (4.24)
until $\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\xi}) := \sum_{t=1}^T \log\left(\sum_{i=1}^K \pi_i \, \mathcal{N}(\boldsymbol{\xi}_t|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\right)$ converges with $\theta \approx \theta^*$
return $\theta^* := \{\pi_i^*, \boldsymbol{\mu}_i^*, \boldsymbol{\Sigma}_i^*\}_{i=1}^K$

Note the variational nature of optimisation where the current estimate of $\boldsymbol{\Sigma}_i^{(\text{diag})}$ is dependent on $\boldsymbol{B}$ and vice versa. Both $\boldsymbol{B}$ and $\boldsymbol{\Sigma}_i^{(\text{diag})}$ are iteratively improved in each EM step and the likelihood is guaranteed to increase at each step.

The mixture components of a semi-tied GMM tend to align themselves towards the basis vectors of $\boldsymbol{H}$. To analyze the impact of this alignment on the encoding of movement synergies, we introduce a *tying factor* $\alpha^{\text{ST}} \in [0, 1]$ that controls the degree of tying of the full covariance matrices with the semi-tied covariance matrices, i.e.,

$$\boldsymbol{\Sigma}_i = \alpha^{\text{ST}} \boldsymbol{H} \boldsymbol{\Sigma}_i^{(\text{diag})} \boldsymbol{H}^\top + (1 - \alpha^{\text{ST}}) \boldsymbol{S}_i, \tag{4.24}$$

where $\alpha^{\text{ST}} = 1$ gives a semi-tied GMM, $\alpha^{\text{ST}} = 0$ leads to a standard GMM, and $(0 < \alpha^{\text{ST}} < 1)$ yields a family of models with intermediate tying of the basis vectors. The overall algorithm is summarized in Alg. 4.

4.2.2 Analysis of Semi-Tied Mixture Models

4.2.2.1 Number of Parameters N_p

The number of parameters for K covariance matrices in semi-tied GMM is smaller than the number of parameters for full covariance matrices in GMM ($D^2 + KD$ compared to $\frac{KD(D+1)}{2}$ of GMM respectively). The decrease in number of parameters is accompanied with additional computational cost of finding $\boldsymbol{B}$ and $\boldsymbol{\Sigma}_i^{(\text{diag})}$ in semi-tied GMM. Compared to semi-tied GMM, standard GMM only requires the estimate of $\boldsymbol{S}_i$ in Eq. (4.22) for the covariance matrix update in each M step. More importantly, semi-tied GMM reveals the latent structure in the data and can be exploited to deal with noisy/insufficient data.

4.2.2.2 Correlation of Mixture Components

To analyse the encoding of semi-tied GMMs, we define $\boldsymbol{M}_c \in \mathbb{R}^{K\times K}$ as the correlation matrix that gives pairwise correlation coefficient between each pair of covariance matrices in the mixture model, i.e.,

$$\boldsymbol{M}_c = \text{corr}\left(\mathbf{vec}(\boldsymbol{\Sigma}_1) \quad \mathbf{vec}(\boldsymbol{\Sigma}_2) \quad \cdots \quad \mathbf{vec}(\boldsymbol{\Sigma}_K)\right), \tag{4.25}$$

where $\mathbf{vec}(\boldsymbol{\Sigma}_i)$ above corresponds to the elements of $\boldsymbol{\Sigma}_i$ in vector form, and $m_c(i,j)$ defines the correlation between the corresponding pair of mixture components. The metric is based on the observation that correlation among the mixture components is higher if they share the same subspace as in semi-tied GMM.

4.2.3 Whole Body Motion Capture Data - Chicken Dance Example

The dataset consists of two subjects performing the chicken dance, publicly available from the CMU motion capture database [Gross 2001]. The dance involves rapid and brisk whole body limb movements with $D = 94$ corresponding to the recorded timestamps ($T \approx 11$ seconds) and the 3-dimensional position of 31 joints for one subject, thereby, making it a challenging problem for the algorithm.

Results of the regenerated dance movement sequence with 75 mixture components and 500 downsampled datapoints are shown in Fig. 4.3. The plots on bottom right show a generic trend where semi-tied GMM ($\alpha^{\text{ST}} = 1$) requires more mixture components to model the training data in comparison to a standard GMM ($\alpha^{\text{ST}} = 0$). Decreasing the tying factor

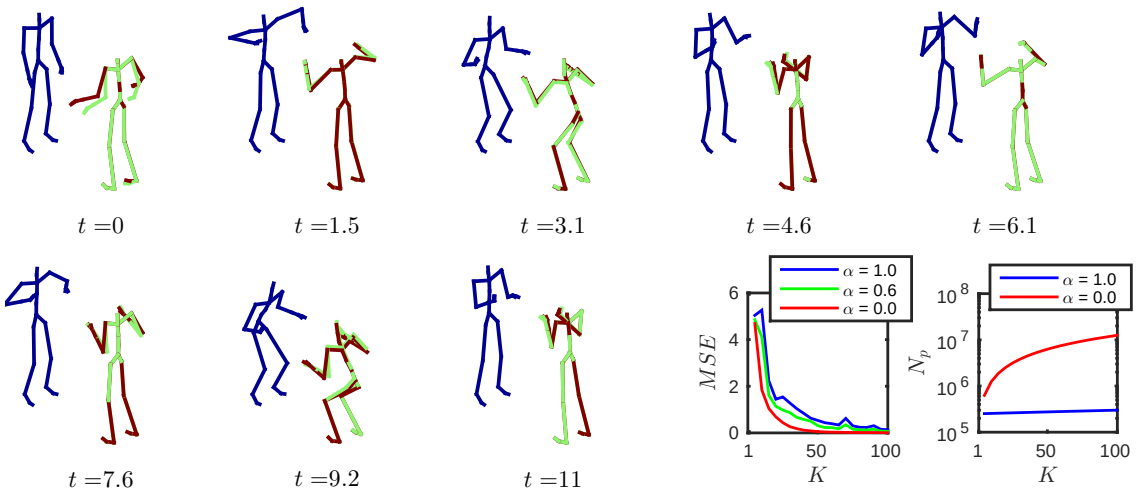


Figure 4.3: Chicken dance movement for the two subjects is shown in blue and red. Regenerated movement for the subject in red is shown in green using Gaussian mixture regression. Two plots on bottom right show comparison of mean squared error (MSE) and the number of parameters N_p of covariance matrix in $\log 10$ scale with increasing number of mixture components K. Time is in seconds, $\alpha = 1$ represents semi-tied GMM, whereas $\alpha = 0$ corresponds to a standard GMM.

in a semi-tied GMM gradually pushes the solution towards a standard GMM as seen with $\alpha^{\text{ST}} = 0.6$ and the resulting MSE curve. The number of parameters, however, remain an order of magnitudes lower for a semi-tied GMM ($15,886$ only in comparison to $334,875$ for a standard GMM with 75 mixture components). Pairwise correlation comparison in Fig. 4.4 reveals that the correlation among the mixture components as defined in Eq. 4.25 increases with the semi-tied GMM in comparison to the correlation observed with the standard GMM.

4.3 Task-Parameterized HSMM in Latent Space

As seen in the previous chapter, task-parameterized models provide a probabilistic formulation to adapt the model parameters for better generalization in new environmental situations. As a quick recap, the demonstrations are observed in P frames of reference defined by the coordinate systems $\{\boldsymbol{A}_j, \boldsymbol{b}_j\}_{j=1}^{P}$. The corresponding hidden state sequence $\{z_t\}_{t=1}^{T}$ with $z_t \in \{1 \dots K\}$ belongs to the discrete set of K cluster indices, $\boldsymbol{a} \in \mathbb{R}^{K \times K}$ with $a_{i,j} \triangleq P(z_t = j | z_{t-1} = i)$ denotes the transition probability of moving from state i to state j, $\{\mu_i^S, \Sigma_i^S\}$ represent the mean and the standard deviation of staying s consecutive steps in state i estimated by a Gaussian $\mathcal{N}(s|\mu_i^S, \Sigma_i^S)$. The hidden state follows a multinomial distribution with $z_t \sim \text{Mult}(\boldsymbol{\pi}_{z_{t-1}})$ where $\boldsymbol{\pi}_{z_{t-1}} \in \mathbb{R}^K$ is the next state transition distribution

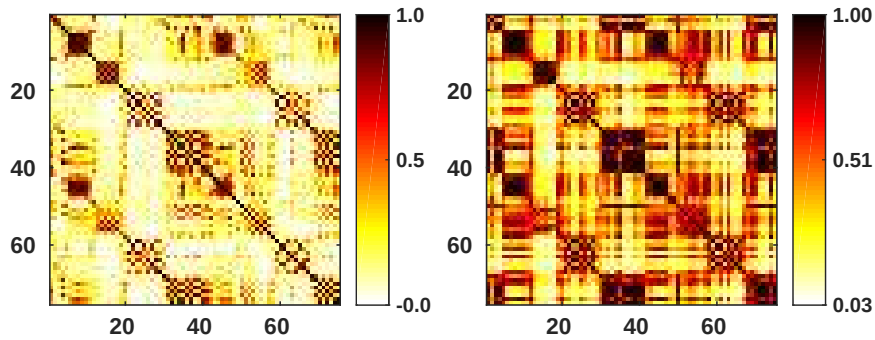


Figure 4.4: Pairwise correlation comparison among the mixture components for whole body motion capture data: *(left)* training with standard GMM, *(right)* training with semi-tied GMM.

over state z_{t-1}, starting from the initial state distribution Π_i. The demonstrations observed from different frames of reference form a third order tensor dataset $\{\boldsymbol{\xi}_t^{(j)}\}_{t,j=1}^{T,P}$ with $\boldsymbol{\xi}_t^{(j)} = \boldsymbol{A}_{t,j}^{-1}(\boldsymbol{\xi}_t - \boldsymbol{b}_{t,j})$. The output distribution of state i in frame j is described by a multivariate Gaussian with parameters $\{\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\}$. The parameters of a task-parameterized HSMM are defined as before, $\theta_h = \left\{\Pi_i, \{a_{i,m}\}_{m=1}^K, \{\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\}_{j=1}^P, \mu_i^S, \Sigma_i^S\right\}_{i=1}^K\}$.

We assume that each Gaussian groups the data in its intrinsic latent space of reduced dimensionality. We use the semi-tied representation of the parameters where the covariance matrices of the mixture model in each frame share a common latent space of the basis vectors $\boldsymbol{H}^{(j)} \in \mathbb{R}^{D\times D}$, and a component specific diagonal matrix $\boldsymbol{\Sigma}_i^{(j)(\text{diag})} \in \mathbb{R}^{D\times D}$ that appropriately maps the subset of basis vectors in the latent space, i.e., $\alpha^{\text{ST}}\boldsymbol{H}^{(j)}\boldsymbol{\Sigma}_i^{(j)(\text{diag})}\boldsymbol{H}^{(j)^\top} + (1-\alpha^{\text{ST}})\boldsymbol{S}_i^{(j)}$ [Tanwani 2016a]. Learning of the model parameters is performed in the same manner as described in Sec. 3.2.1.2, except the latent space parameters $\{\boldsymbol{H}^{(j)}, \boldsymbol{\Sigma}_i^{(j)(\text{diag})}\}$ are estimated as described in Alg. 4 for each frame. Increasing α^{ST} from 0 to 1 increases the effect of tying the mixture components in the task-parameterized formulation. Note that the generalization to MFA decomposition is straightforward by setting $\boldsymbol{\Sigma}_i^{(j)} = \boldsymbol{\Lambda}_i^{(j)}\boldsymbol{\Lambda}_i^{(j)^\top} + \boldsymbol{\Psi}_i^{(j)}$, and estimating the latent space parameters as described in Eq. (4.9) and Eq. (4.10) for each frame respectively.

For a new environmental situation represented by the frames $\{\tilde{\boldsymbol{A}}_j, \tilde{\boldsymbol{b}}_j\}_{j=1}^P$, the resulting model parameters $\{\tilde{\boldsymbol{\mu}}_i, \tilde{\boldsymbol{\Sigma}}_i\}$ are obtained by first linearly transforming the Gaussians in the P frames with

$$\mathcal{N}(\tilde{\boldsymbol{\mu}}_i^{(j)}, \tilde{\boldsymbol{\Sigma}}_i^{(j)}) = \mathcal{N}\left(\tilde{\boldsymbol{A}}_j\boldsymbol{\mu}_i^{(j)} + \tilde{\boldsymbol{b}}_j, \tilde{\boldsymbol{A}}_j\boldsymbol{\Sigma}_i^{(j)}\tilde{\boldsymbol{A}}_j^\top\right), \tag{4.26}$$

and then computing the products of the linearly transformed Gaussians for each component

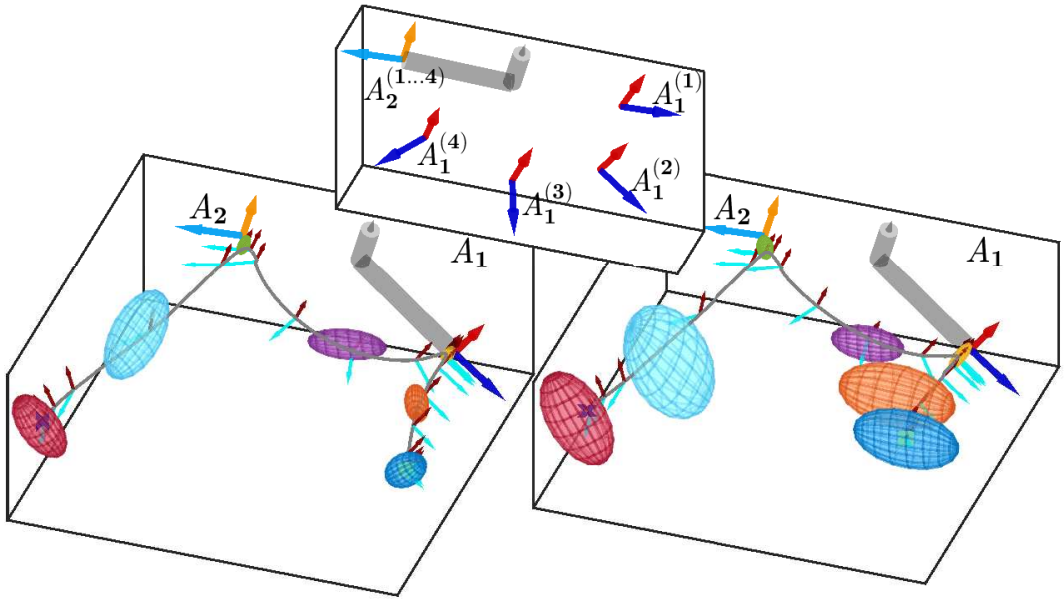


Figure 4.5: Baxter valve opening movement reproduction for an unseen valve configuration: *(left)* encoding with task-parameterized HSMM ($\alpha^{\text{ST}} = 0$), *(right)* encoding with task parameterized semi-tied HSMM ($\alpha^{\text{ST}} = 1$). Note that the mixture components are better aligned and scaled in task-parameterized semi-tied HSMM than task-parameterized HSMM with full covariance matrices.

with

$$\mathcal{N}(\tilde{\boldsymbol{\mu}}_i, \tilde{\boldsymbol{\Sigma}}_i) \propto \prod_{j=1}^{P} \mathcal{N}(\tilde{\boldsymbol{\mu}}_i^{(j)}, \tilde{\boldsymbol{\Sigma}}_i^{(j)}), \tag{4.27}$$

$$\tilde{\boldsymbol{\Sigma}}_i = \left(\sum_{j=1}^{P} \tilde{\boldsymbol{\Sigma}}_i^{(j)} \right)^{-1} \tilde{\boldsymbol{\mu}}_i = \tilde{\boldsymbol{\Sigma}}_i \sum_{j=1}^{P} \left(\tilde{\boldsymbol{\Sigma}}_i^{(j)} \right)^{-1} \left(\tilde{\boldsymbol{\mu}}_i^{(j)} \right).$$

Note that the latent space dimension of the product of Gaussians is defined by the minimum of corresponding subspace dimensions of the Gaussians in P frames, i.e., if the latent space dimension of Gaussians in each frame is the same, the latent space dimension of the resulting product of Gaussians is also the same. The degenerate Gaussians signify the important directions in the demonstrations along which the movement is constrained during reproduction. The adapted model parameters in a new situation are used to retrieve a smooth trajectory with linear quadratic tracking as shown in the previous chapter.

4.3.1 Valve Opening Comparison

In the previous chapter, we introduced the valve opening task using the torque-controlled Baxter robot. Here, we compare its performance with the task-parameterized semi-tied HSMM for exploiting the coordination patterns and reusing the synergistic directions such as when reaching the valve and when coming back to a neutral joint angle configuration (home position).

Results of the regenerated movements with 7 mixture components for task-parameterized HSMM with and without semi-tied parameters are shown in Fig. 4.5. It can be seen that the semi-tied mixture components are better aligned and scaled, while independently modeling each covariance matrix is prone to over-fitting. Semi-tying model parameters allows encoding of similar coordination patterns with a set of basis vectors. The alignment of correlated mixture components improves the generalization ability of the model in previously unseen situations.

Table 4.1 quantifies the encoding results with different values of α^{ST}. We can see that the task-parameterized semi-tied HSMM ($\alpha^{\text{ST}} = 1$) drastically reduces the number of parameters and yields better testing error than training error compared to task-parameterized HSMM with $\alpha^{\text{ST}} = 0$.

Table 4.1: Performance analysis of tying factor α^{ST} in task-parameterized semi-tied HSMM with training MSE, testing MSE, number of covariance matrix parameters using 7 mixture components and 2 frames, and time required for training the model in seconds. Units are in meters.

α^{ST}	**Training MSE**	**Testing MSE**	**Number of Parameters**	**Training Time (s)**
valve opening				
0.0	**0.0021**	0.0146	1470	**2.45**
0.5	0.0038	0.0119	1470	5.40
1.0	0.0040	**0.0119**	**588**	9.78
pick-and-place via obstacle avoidance				
0.0	**0.0023**	0.0138	1470	**2.21**
0.5	0.0028	0.0129	1470	4.73
1.0	0.0033	**0.0127**	**588**	10.21

4.3.2 Pick-and-Place with Obstacle Avoidance Example

The objective in this task is to place the object in a desired target position by picking it from different initial positions and orientations of the object, while adapting the movement

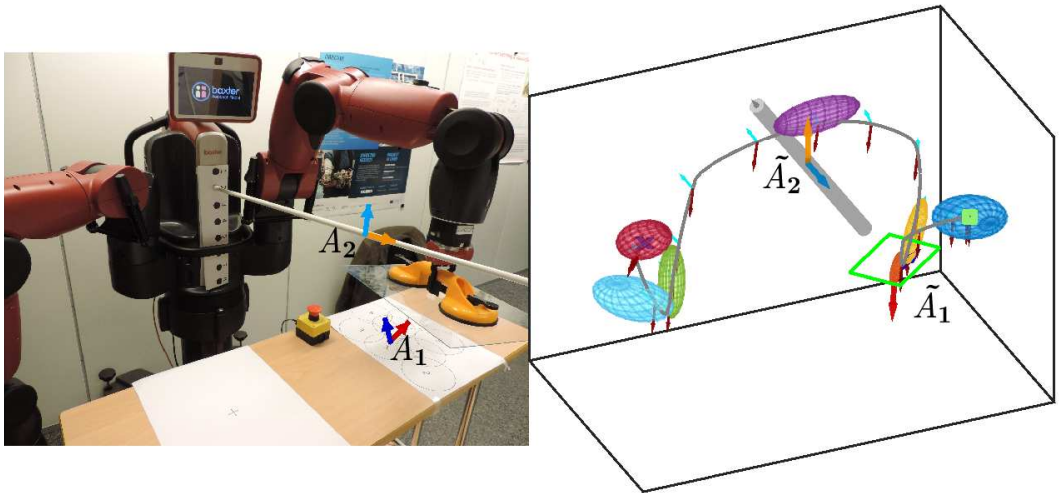


Figure 4.6: *(left)* Baxter robot picks the glass plate with a suction lever and places it on the cross after avoiding an obstacle of varying height, *(right)* reproduction for previously unseen object and obstacle position.

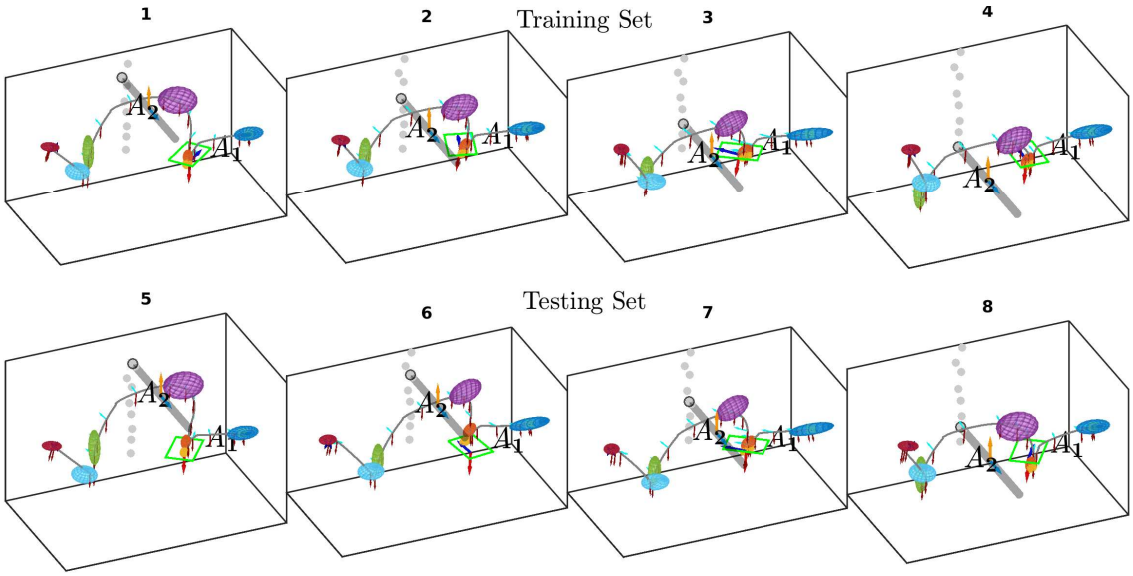


Figure 4.7: Task-Parameterized Semi-Tied HSMM performance on pick-and-place with obstacle avoidance task: *(top)* training set reproductions, *(bottom)* testing set reproductions.

to avoid the obstacle. The setup of pick-and-place task with obstacle avoidance is shown in Fig. 4.7. The Baxter robot is required to grasp the glass plate with a suction lever placed in an initial configuration as marked on the setup. The obstacle can be vertically displaced to one of the 8 target configurations. We describe the task with two frames, one for the object initial configuration with $\{\boldsymbol{A}_1, \boldsymbol{b}_1\}$ as defined in Eq. (3.57) and other for the obstacle $\{\boldsymbol{A}_2, \boldsymbol{b}_2\}$ with $\boldsymbol{A}_2 = \boldsymbol{I}$ and $\boldsymbol{b}_2$ to specify the centre of the obstacle. We collect 8 kinesthetic demonstrations with different initial configurations of the object and the obstacle successively displaced upwards as marked with the visual tags in the figure. Alternate demonstrations $\{1, 3, 5, 7\}$ are used for the training set, while the rest are used for the test set. Each observation comprises of the end-effector Cartesian position $\boldsymbol{x}_t^p \in \mathbb{R}^3$, quaternion orientation $\boldsymbol{\varepsilon}_t^o \in \mathbb{R}^4$, linear velocity $\dot{\boldsymbol{x}}_t^p \in \mathbb{R}^3$, and quaternion derivative $\dot{\boldsymbol{\varepsilon}}_t^o \in \mathbb{R}^4$ with $\boldsymbol{\xi}_t = \left[\boldsymbol{x}_t^{p\top}\ \boldsymbol{\varepsilon}_t^{o\top}\ \dot{\boldsymbol{x}}_t^{p\top}\ \dot{\boldsymbol{\varepsilon}}_t^{o\top}\right]^\top$, $D = 14, P = 2$, and a total of 200 datapoints per demonstration.

During evaluation of the learned task-parameterized semi-tied HSMM, we can see that the robot arm is able to generalize effectively by following a similar pattern to the recorded demonstrations in picking and placing the object (see Fig. 4.7 for reproductions). The model even proves robust to the examples requiring extrapolation of the training data. Table 4.1 depicts a similar trend to the valve opening task, thereby verifying the efficacy of the proposed method for learning manipulation tasks.

4.4 Conclusion

In this chapter, we have presented generative models in latent space for robust learning and adaptation of robot manipulation tasks. We have presented a technique to tie the covariance matrices of the mixture model with a shared set of basis vectors. The approach is based on the hypothesis that similar coordination patterns occur at different phases in a manipulation task. By exploiting the spatial and temporal correlation in the demonstrations, we reduced the number of parameters to be estimated while locking the most important synergies to cope with perturbations. This allowed the reuse of the discovered synergies in different parts of the task having similar coordination patterns. In contrast, the MFA decomposition of each covariance matrix separately cannot exploit the temporal synergies, and has more flexibility in locally encoding the data. Recently, semi-tying model parameters has also been shown to explain multiple movements in generating calligraphic movements [Berio 2017]. We have shown that the task-parameterized semi-tied HSMM encoding enables the robot to autonomously deal with different situations in manipulation tasks with much fewer parameters and better generalization ability. This has enabled the

Baxter robot to tackle valve opening and pick-and-place via obstacle avoidance problems from previously unseen configurations of the environment.

Chapter 5

Bayesian Non-Parametric Online Generative Models

Contents

5.1 Background and Related Work 99
5.2 Problem Formulation under Small Variance Asymptotics (SVA) . 101
5.3 SVA of DP-GMM 103
5.3.1 Dirichlet Process GMM (DP-GMM) 103
5.3.2 Online Inference in DP-GMM 104
5.4 Online DP-MPPCA 105
5.4.1 Dirichlet Process MPPCA (DP-MPPCA) 105
5.4.2 Online Inference in DP-MPPCA 107
5.5 Online HDP-HSMM 111
5.5.1 Hierarchical Dirichlet Process HSMM (HDP-HSMM) 111
5.5.2 Online Inference in HDP-HSMM 113
5.6 Scalable Online Sequence Clustering (SOSC) 115
5.6.1 Task-Parameterized Formulation of SOSC 117
5.7 Experiments, Results and Discussions 119
5.7.1 Synthetic Data 120
5.7.2 Tracking Screwdriver Target and Hooking Carabiner Examples 122
5.8 Conclusions 128

Adapting statistical learning models online with large scale streaming data is a challenging problem. Bayesian non-parametric mixture models provide flexibility in model selec-

tion, however, their widespread use is limited by the computational overhead of existing sampling-based and variational techniques for inference. Small variance asymptotics is emerging as a useful technique for inference in large scale Bayesian non-parametric mixture models. This chapter analyses the online learning of robot manipulation tasks with Bayesian non-parametric mixture models under small variance asymptotics [Tanwani 2016c, Tanwani 2016b]. The analysis gives a *scalable online sequence clustering* (SOSC) algorithm that is non-parametric in the number of clusters and the subspace dimension of each cluster. SOSC groups the new datapoint in its low dimensional subspace by online inference in a non-parametric *mixture of probabilistic principal component analyzers* (MPPCA) based on Dirichlet process, and captures the state transition and state duration information online in a *hidden semi-Markov model* (HSMM) based on hierarchical Dirichlet process. Task-parameterized formulation of our approach autonomously adapts the model to changing environmental situations during manipulation. We apply the algorithm in a teleoperation setting to recognize the intention of the operator and remotely adjust the movement of the robot using the learned model. The generative model is used to synthesize both time-independent and time-dependent behaviours by relying on the principles of shared and autonomous control. Experiments with the Baxter robot yield parsimonious clusters that adapt online with new demonstrations and assist the operator in performing remote manipulation tasks.

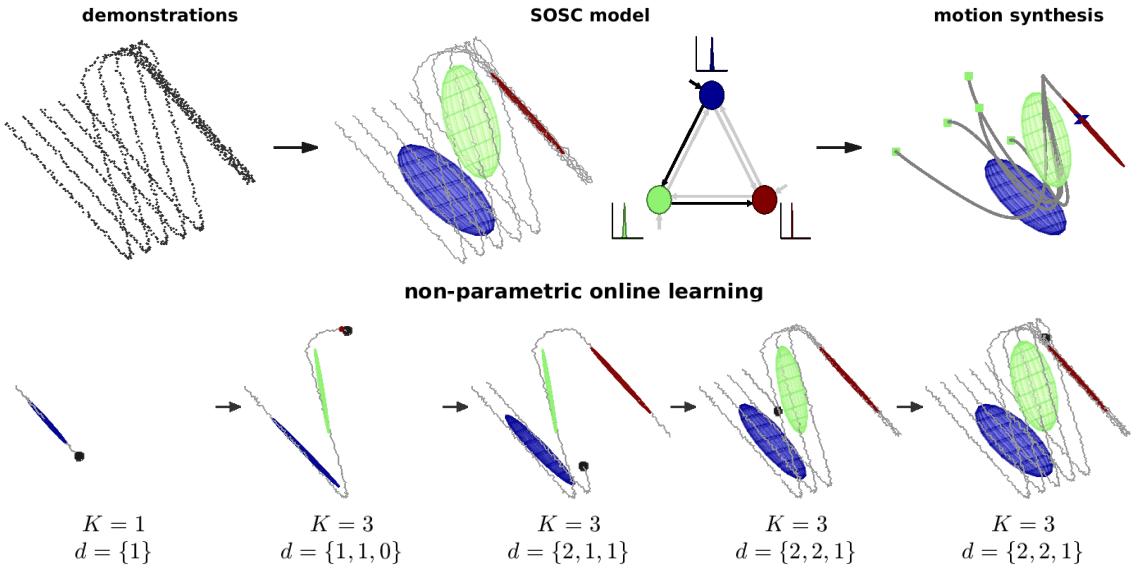


Figure 5.1: SOSC model illustration with Z-shaped streaming data composed of multiple trajectory samples. The model incrementally clusters the data in its intrinsic subspace. It tracks the transition among states and the state duration steps in a non-parametric manner. The generative model is used to recognize and synthesize motion in performing robot manipulation tasks.

5.1 Background and Related Work

With the influx of high-dimensional sensory data in robotics, an open challenge is to compactly encode the data online so that the robots are able to perform under varying environmental situations and across range of different tasks. **Online/Incremental learning** methods update the model parameters with streaming data, without the need to re-train the model in a batch manner [Neal 1999, Song 2005]. Incremental online learning poses a unique challenge to the existing robot learning methods with high-dimensional data, model selection, real-time adaptation and adequate accuracy or generalization after observing a fewer number of training samples. Non-parametric regression methods have been commonly used in this context such as locally weighted projection regression [Vijayakumar 2005], sparse online Gaussian process regression [Gijsberts 2013] and their fusion with local Gaussian process regression [Nguyen-Tuong 2009]. Kulic et al. used HMMs to incrementally group whole-body motions based on their relative distance in HMM space [Kulic 2008]. Lee and Ott presented an iterative motion primitive refinement approach with HMMs [Lee 2010a]. Kronander et al. locally reshaped an existing dynamical system with new demonstrations in an incremental manner while preserving its stability [Kronander 2015]. Hoyos et al. experimented with different strategies to incrementally add demonstrations to a task-parametrized GMM [Hoyos 2016]. Bruno et al. learned autonomous behaviours for a flexible surgical robot by online clustering with DP-means [Bruno 2016].

Bayesian non-parametric treatment of the HMMs/HSMMs provides flexibility in model selection by maintaining an appropriate probability distribution over parameter values, $\mathcal{P}(\boldsymbol{\xi}_t) = \int \mathcal{P}(\boldsymbol{\xi}_t|\theta)\mathcal{P}(\theta)d\theta$. They automate the number of states selection procedure by Bayesian inference in a model with infinite number of states [Beal 2002, Johnson 2013]. Niekum et al. used the Beta Process Autoregressive HMM for learning from unstructured demonstrations [Niekum 2012]. The authors in [Garg 2016, Krishnan 2018] defined a hierarchical non-parametric Bayesian model to identify the transition structure between states with a linear dynamical system. Figueroa et al. used the transformation invariant covariance matrix for encoding tasks with a Bayesian non-parametric HMM [Figueroa 2017]. Inferring the maximum *a posteriori* distribution of the parameters in non-parametric models, however, is often difficult. Markov Chain Monte Carlo (MCMC) sampling or variational methods are required which are difficult to implement and often do not scale with the size of the data. Although attractive for encapsulating *a priori* information about the task, the computational overhead of existing sampling-based and variational techniques for inference limit the widespread use of these models.

Recent analysis of Bayesian non-parametric mixture models under *small variance asymptotic* (SVA) limit has led to simple deterministic models that scale well with large size applications. For example, as the variances of the mixture model tend to zero in a GMM, the probabilistic model converges to its deterministic counterpart, k-means, or to its non-parametric Dirichlet process (DP) version, DP-Means [Kulis 2012]. SVA analysis of other richer probabilistic models such as dependent DP mixture models [Campbell 2013], hierarchical Dirichlet process (HDP) [Jiang 2012], infinite latent feature models [Broderick 2013], Markov jump processes [Huggins 2015], infinite hidden Markov models [Roychowdhury 2013], and infinite mixture of probabilistic principal component analysers (MPPCA) [Wang 2015] leads to similar algorithms that scale well and yet retain the flexibility of non-parametric models.

This chapter builds upon these advancements in small variance asymptotic analysis of Bayesian non-parametric mixture models. We present a non-parametric online unsupervised framework for robot learning from demonstrations, which scales well with sequential high-dimensional data. We formulate online inference algorithms of DP-GMM, DP-MPPCA, and HDP-HSMM under small variance asymptotics. We then learn a task-parameterized generative model online for encoding and motion synthesis of robot manipulation tasks.

In this chapter, we seek to incrementally update the parameters θ with each new observation $\boldsymbol{\xi}_{t+1}$ without having to retrain the model in a batch manner and store the demonstration data. We present an online inference algorithm for clustering sequential data, called scalable online sequence clustering (SOSC). SOSC incrementally groups the streaming data in its low-dimensional subspace by online inference in the Dirichlet process MPPCA under small variance asymptotics, while being non-parametric in the number of clusters and the subspace dimension of each cluster. The model encapsulates the state transition and the state duration information in the data with online inference in an HSMM based on HDP. A task-parameterized formulation of the model is used to adapt the model parameters to varying environmental situations in a probabilistic manner [Tanwani 2016a]. The proposed approach uses the learning from demonstrations paradigm to teach manipulation tasks to robots in an online and intuitive manner. We show its application in a teleoperation scenario where the SOSC model is built online from the demonstrations provided by the teleoperator to perform remote robot manipulation tasks (see Fig. 5.1 for an overview of our approach). The chapter contains the following,

- Online inference algorithms for DP-GMM, DP-MPPCA and HDP-HSMM under small variance asymptotics,

- Non-parametric SOSC algorithm for online learning and motion synthesis of high-dimensional robot manipulation tasks,
- Task-parameterized formulation of the SOSC model to systematically adapt the model parameters to changing situations such as position/orientation/size of the objects,
- Learning manipulation tasks from demonstrations for semi-autonomous teleoperation.

5.2 Problem Formulation under Small Variance Asymptotics (SVA)

Let us consider the streaming observation sequence $\{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_t\}$ with $\boldsymbol{\xi}_t \in \mathbb{R}^D$ obtained at current time step t while demonstrating a manipulation task. The corresponding hidden state sequence $\{z_1, \ldots, z_t\}$ with $z_t \in \{1, \ldots, K\}$ belongs to the discrete set of K cluster indices at time t, and the observation $\boldsymbol{\xi}_t$ is drawn from a multivariate Gaussian with mixture coefficients $\pi_{t,i} \in \mathbb{R}$, mean $\boldsymbol{\mu}_{t,i} \in \mathbb{R}^D$ and covariance $\boldsymbol{\Sigma}_{t,i} \in \mathbb{R}^{D \times D}$ at time t.

We seek to update the parameters online upon observation of a new datapoint $\boldsymbol{\xi}_{t+1}$, such that the datapoint can be discarded afterwards. Small variance asymptotic (SVA) analysis implies that the covariance matrix $\boldsymbol{\Sigma}_{t,i}$ of all the Gaussians reduces to the isotropic noise σ^2, i.e., $\boldsymbol{\Sigma}_{t,i} \approx \lim_{\sigma^2 \to 0} \sigma^2 \boldsymbol{I}$ [Kulis 2012, Broderick 2013, Roychowdhury 2013]. Note that if the covariance matrices $\boldsymbol{\Sigma}_{t,i}$ of all the mixture components in a GMM are set equal to the isotropic matrix $\sigma^2 \boldsymbol{I}$, the expected value of the complete log-likelihood of the data a.k.a. the auxiliary function, $\mathcal{Q}(\Theta_{\text{GMM}}, \Theta_{\text{GMM}}^{\text{old}}) = \mathbb{E}\left\{\log \mathcal{P}(\boldsymbol{\xi}_t, z_t | \Theta_{\text{GMM}}) \mid \boldsymbol{\xi}_t, \Theta_{\text{GMM}}^{\text{old}}\right\}$, takes the form [Dempster 1977]

$$\sum_{i=1}^{K} \mathcal{P}(i|\boldsymbol{\xi}_t, \Theta_{\text{GMM}}^{\text{old}}) \left(\log \pi_{t,i} - \frac{D}{2} \log 2\pi\sigma^2 - \frac{\|\boldsymbol{\xi}_t - \boldsymbol{\mu}_{t,i}\|_2^2}{2\sigma^2} \right). \tag{5.1}$$

Applying the small variance asymptotic limit to the auxiliary function with $\lim_{\sigma^2 \to 0} \mathcal{Q}(\Theta_{\text{GMM}}, \Theta_{\text{GMM}}^{\text{old}})$, the last term $\frac{\|\boldsymbol{\xi}_t - \boldsymbol{\mu}_{t,i}\|_2^2}{2\sigma^2}$ dominates the objective function and the maximum likelihood estimate reduces to the k-means problem,[1] i.e.,

$$\max \mathcal{Q}(\Theta_{\text{GMM}}, \Theta_{\text{GMM}}^{\text{old}}) = \underset{z_t, \boldsymbol{\mu}_t}{\arg\min} \|\boldsymbol{\xi}_t - \boldsymbol{\mu}_{t,z_t}\|_2^2. \tag{5.2}$$

[1] SVA analysis of the Bayesian non-parametric GMM leads to the DP-means algorithm [Kulis 2012]. Similarly, SVA analysis of the HMM yields the segmental k-means problem [Roychowdhury 2013].

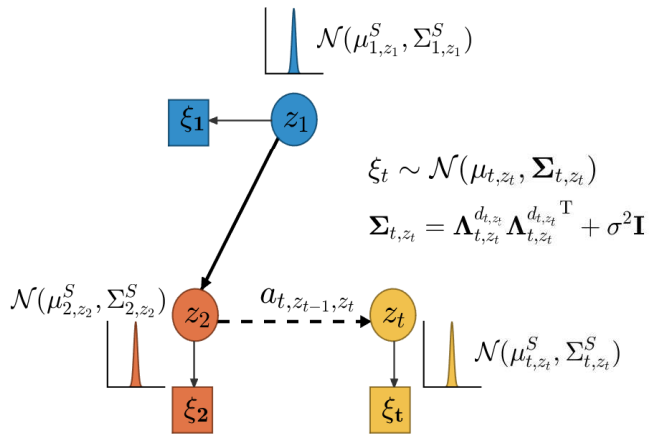


Figure 5.2: SOSC parameter representation using non-parametric HSMM with non-parameter MPPCA as observation distribution given the streaming data $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots \boldsymbol{\xi}_t$.

By restricting the covariance matrix to an isotropic/spherical noise, the number of parameters grows up to a constant with the dimension of datapoint D. Although attractive for scalability and parsimonious structure, such decoupling cannot encode the important motor control principles of coordination, synergies and action-perception couplings as we have seen in the previous chapter. Consequently, we further assume that the ith output Gaussian groups the observation $\boldsymbol{\xi}_t$ in its intrinsic low-dimensional affine subspace of dimension $d_{t,i}$ at time t with projection matrix $\mathbf{\Lambda}^{d_{t,i}}_{t,i} \in \mathbb{R}^{D \times d_{t,i}}$, such that $d_{t,i} < D$ and $\mathbf{\Sigma}_{t,i} = \mathbf{\Lambda}^{d_{t,i}}_{t,i} \mathbf{\Lambda}^{d_{t,i}\top}_{t,i} + \sigma^2 \boldsymbol{I}$. Under this assumption, we apply the small variance asymptotic limit on the remaining $(D - d_{t,i})$ dimensions to encode the most important coordination patterns while being parsimonious in the number of parameters.

In order to encode the temporal information among the mixture components, let $\boldsymbol{a}_t \in \mathbb{R}^{K \times K}$ with $a_{t,i,j} \triangleq P(z_t = j | z_{t-1} = i)$ denote the transition probability of moving from state i at time $t-1$ to state j at time t. The parameters $\{\mu^S_{t,i}, \Sigma^S_{t,i}\}$ represent the mean and the standard deviation of staying s consecutive time steps in state i estimated by a Gaussian $\mathcal{N}(s | \mu^S_{t,i}, \Sigma^S_{t,i})$. The hidden state follows a multinomial distribution with $z_t \sim \mathrm{Mult}(\boldsymbol{\pi}_{z_{t-1}})$ where $\boldsymbol{\pi}_{z_{t-1}} \in \mathbb{R}^K$ is the next state transition distribution over state z_{t-1}, and the observation $\boldsymbol{\xi}_t$ is drawn from the output distribution of state j, described by a multivariate Gaussian with parameters $\{\boldsymbol{\mu}_{t,j}, \mathbf{\Sigma}_{t,j}\}$ (see Fig. 5.2 for graphical representation of the problem). The K Gaussian components constitute a GMM augmented with the state transition and the state duration model to capture the sequential pattern in the demonstrations.

The overall parameter set of SOSC is represented by $\Theta_{t,\mathrm{SOSC}} =$

$\left\{\boldsymbol{\mu}_{t,i}, \boldsymbol{\Sigma}_{t,i}, \{a_{t,i,m}\}_{m=1}^{K}, \mu_{t,i}^{S}, \Sigma_{t,i}^{S}\right\}_{i=1}^{K}$.[2] We are interested in updating the parameter set $\Theta_{t,\text{SOSC}}$ online upon observation of a new datapoint $\boldsymbol{\xi}_{t+1}$, such that the datapoint can be discarded afterwards. We first apply the Bayesian non-parametric treatment to the underlying mixture models and formulate online inference algorithms for DP-MPPCA and HDP-HSMM under small variance asymptotics. This results in a non-parametric online approach to robot learning from demonstrations.

5.3 SVA of DP-GMM

In this section, we review the fundamentals of Bayesian non-parametric extension of GMM under small variance asymptotics using the parameter subset $\Theta_{\text{GMM}} = \{\pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{K}$ and present a simple approach for online update of the parameters.

5.3.1 Dirichlet Process GMM (DP-GMM)

Consider a Bayesian non-parametric GMM with *Chinese Restaurant Process* (CRP) prior over the cluster assignment with α^{DP} as concentration parameter, $z_t \sim \text{CRP}(\alpha^{\text{DP}})$, and non-informative prior over cluster means with ϱ^2 as small constant, $\boldsymbol{\mu}_i \sim \mathcal{N}(\mathbf{0}, \varrho^2 \boldsymbol{I}_D)$. The likelihood function for a set of datapoints is evaluated as

$$\mathcal{P}(\boldsymbol{\xi}_t|\boldsymbol{z}, \boldsymbol{\mu}) = \prod_{i=1}^{K}\prod_{t=1}^{T} \mathcal{N}(\boldsymbol{\xi}_t|\boldsymbol{\mu}_i, \sigma^2\boldsymbol{I}). \tag{5.3}$$

The parameters $\boldsymbol{z}$ and $\boldsymbol{\mu}$ are obtained by maximizing the posterior distribution

$$\underset{K,\boldsymbol{z},\boldsymbol{u}}{\arg\max}\, \mathcal{P}(\boldsymbol{z}, \boldsymbol{\mu}|\boldsymbol{\xi}_t) \propto \underset{K,\boldsymbol{z},\boldsymbol{u}}{\arg\min} - \log \mathcal{P}(\boldsymbol{\xi}_t, \boldsymbol{z}, \boldsymbol{\mu}). \tag{5.4}$$

Computing the joint posterior distribution and setting $\alpha^{\text{DP}} = \exp(-\frac{\lambda}{2\sigma^2})$

$$\mathcal{P}(\boldsymbol{\xi}_t, \boldsymbol{z}, \boldsymbol{\mu}) = \mathcal{P}(\boldsymbol{\xi}_t|\boldsymbol{z}, \boldsymbol{\mu})\, \mathcal{P}(\boldsymbol{z})\, \mathcal{P}(\boldsymbol{\mu}) = \prod_{i=1}^{K}\prod_{t=1}^{T} \mathcal{N}(\boldsymbol{\xi}_t|\boldsymbol{\mu}_i, \sigma^2\boldsymbol{I})\, \text{CRP}(\exp(-\frac{\lambda}{2\sigma^2}))\, \mathcal{N}(\mathbf{0}, \varrho^2\boldsymbol{I}_D). \tag{5.5}$$

Taking the log of the joint posterior distribution and applying the SVA limit $\lim_{\sigma^2\to 0}$ yields the DP-means algorithm [Kulis 2012]. The limit pushes the posterior mass on one of the clusters leading to a deterministic assignment based on the distance of the datapoint to

[2] With a slight abuse of notation, we represent the parameters with an added subscript t for online learning. For example, $\Theta_{t,h}$ denotes the parameters of Θ_h at time t.

the nearest cluster. The resulting loss function $\mathcal{L}(\boldsymbol{z}, \boldsymbol{\mu})$ to optimize is given as

$$\underset{K,\boldsymbol{z},\boldsymbol{u}}{\arg\min} \lim_{\sigma^2 \to 0} -\log \mathcal{P}(\boldsymbol{\xi}_t, \boldsymbol{z}, \boldsymbol{\mu}) \approx \underset{K,\boldsymbol{z},\boldsymbol{u}}{\arg\min} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\mu}) = \underset{K,\boldsymbol{z},\boldsymbol{u}}{\arg\min} \sum_{i=1}^{K} \sum_{t=1}^{T} \|\boldsymbol{\xi}_t - \boldsymbol{\mu}_i\|_2^2 + \lambda K. \quad (5.6)$$

The algorithm is similar to k-means algorithm except that it is non-parametric in the number of clusters. The algorithm iteratively assigns the datapoint(s) to its nearest cluster center, and if any of the datapoints are farther away from the cluster center than a certain threshold λ, a new cluster is created with the distant datapoints and a penalty λ added to the loss function. The algorithm converges to a local minimum just like the k-means algorithm.

5.3.2 Online Inference in DP-GMM

In the online setting, we want to update the parameters $\Theta_{t,\text{GMM}}$ with each new observation $\boldsymbol{\xi}_{t+1}$ such that the loss function in Eq. (5.6) is minimized. The update consists of the cluster assignment step and incremental update of parameters step.

5.3.2.1 Cluster Assignment z_{t+1}:

In the online setting, the cluster assignment z_{t+1} for new datapoint $\boldsymbol{\xi}_{t+1}$ is based on the distance of the datapoint to the existing cluster means. If the minimum distance is greater than a certain threshold λ, a new cluster is initialized with that datapoint; otherwise the assigned cluster prior, mean and the corresponding number of datapoints $w_{t+1,z_{t+1}}$ are incrementally updated. We can thus write,

$$z_{t+1} = \underset{j=1\ldots K+1}{\arg\min} \begin{cases} \|\boldsymbol{\xi}_{t+1} - \boldsymbol{\mu}_{t,j}\|_2^2, & \text{if } j \leq K \\ \lambda, & \text{otherwise.} \end{cases} \quad (5.7)$$

5.3.2.2 Parameters Update $\Theta_{t+1,\mathbf{GMM}}$:

Given the cluster assignment $z_{t+1} = i$ and the covariance matrix set to $\boldsymbol{\Sigma}_{t,i} = \sigma^2 \boldsymbol{I}$, the parameters are updated with

$$\pi_{t+1,i} = \frac{1}{t+1}\Big(t\pi_{t,i} + 1\Big),$$
$$\boldsymbol{\mu}_{t+1,i} = \frac{1}{w_{t,i}+1}\Big(w_{t,i}\boldsymbol{\mu}_{t,i} + \boldsymbol{\xi}_{t+1}\Big), \tag{5.8}$$

where $w_{t,i}$ is the weight assigned to the i-th cluster parameter set at time t to control the effect of the parameter update with the new datapoint at time $t+1$ relative to the updates seen till time t (see next section for updates of $w_{t+1,i}$).

Loss function $\mathcal{L}(z_{t+1}, \boldsymbol{\mu}_{t+1,z_{t+1}})$: The loss function optimized at time step $t+1$ is

$$\mathcal{L}(z_{t+1}, \boldsymbol{\mu}_{t+1,z_{t+1}}) = \lambda K + \|\boldsymbol{\xi}_{t+1} - \boldsymbol{\mu}_{t+1,z_{t+1}}\|_2^2 \leq \mathcal{L}(z_{t+1}, \boldsymbol{\mu}_{t,z_{t+1}}). \tag{5.9}$$

It can be seen that direct application of small variance asymptotic limit with isotropic Gaussians severely limits the model from encoding important coordination patterns/variance in the streaming data. We next apply the limit to discard only the redundant dimensions in a non-parametric manner and project the new datapoint in a latent subspace by online inference in a Dirichlet process mixture of probabilistic principal component analyzers.

5.4 Online DP-MPPCA

In this section, we consider the problem formulation with a *mixture of probabilistic principal component analyzers* (MPPCA) using the parameter subset $\Theta_{\text{MPPCA}} = \{\boldsymbol{\mu}_i, \boldsymbol{\Lambda}_i^d, d_i\}_{i=1}^K$. We consider its non-parametric extension with the Dirichlet process under small variance asymptotics and present an algorithm for online inference.

5.4.1 Dirichlet Process MPPCA (DP-MPPCA)

The basic idea of MPPCA is to reduce the dimensions of the data while keeping the observed covariance structure. The generative model of MPPCA approximates the datapoint $\boldsymbol{\xi}_t$ as a convex combination of K subspace clusters [Tipping 1999]

$$\mathcal{P}(\boldsymbol{\xi}_t|\Theta_{\text{MPPCA}}) = \sum_{i=1}^{K} \mathcal{P}(z_t = i)\, \mathcal{N}(\boldsymbol{\xi}_t|\boldsymbol{\mu}_i, \boldsymbol{\Lambda}_i^d \boldsymbol{\Lambda}_i^{d^\top} + \sigma_i^2 \boldsymbol{I}), \tag{5.10}$$

where $\mathcal{P}(z_t = i)$ is the cluster prior, $\boldsymbol{\Lambda}_i^d \in \mathbb{R}^{D\times d}$ is the projection matrix with $d < D$ and $d = d_i$, $\sigma_i^2 \boldsymbol{I}$ is the isotropic noise coefficient for the i-th cluster, and the covariance structure is of the form $\boldsymbol{\Sigma}_i = \boldsymbol{\Lambda}_i^d \boldsymbol{\Lambda}_i^{d^\top} + \sigma_i^2 \boldsymbol{I}$.[3] The model assumes that $\boldsymbol{\xi}_t$, conditioned on $z_t = i$, is generated by an affine transformation of d-dimensional latent variable $\boldsymbol{f}_t \in \mathbb{R}^d$ with noise term $\varepsilon \in \mathbb{R}^D$ such that

$$\boldsymbol{\xi}_t = \boldsymbol{\Lambda}_i^d \boldsymbol{f}_t + \boldsymbol{\mu}_i + \boldsymbol{\varepsilon}, \quad \boldsymbol{f}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d), \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma_i^2 \boldsymbol{I}). \tag{5.11}$$

The model parameters of MPPCA are usually learned using an Expectation-Maximization (EM) procedure [Tipping 1999]. But in this case, both the number of clusters K and the subspace dimension of each cluster d need to be specified a priori, which is not always trivial in several domains.

Bayesian non-parametric extension of MPPCA alleviates the problem of model selection by defining prior distributions over the number of clusters K and the subspace dimension of each cluster d_i [Zhang 2004, Chen 2010, Wang 2015]. Similar to DP-GMM, a CRP prior is placed over the cluster assignment $z_t \sim \mathrm{CRP}(\alpha^{\mathrm{DP}})$, along with a hierarchical prior over the projection matrix $\boldsymbol{\Lambda}_i^{d_i}$ and an exponential prior on the subspace rank $d_i \sim r^{d_i}$ where $r \in (0, 1)$. Applying small variance asymptotics on the resulting partially collapsed Gibbs sampler leads to an efficient deterministic algorithm for subspace clustering with an infinite MPPCA [Wang 2015]. The algorithm iteratively converges by minimizing the loss function

$$\mathcal{L}(\boldsymbol{z}, \boldsymbol{d}, \boldsymbol{\mu}, \boldsymbol{U}) = \lambda K + \lambda_1 \sum_{i=1}^{K} d_i + \sum_{t=1}^{T} \mathrm{dist}(\boldsymbol{\xi}_t, \boldsymbol{\mu}_{z_t}, \boldsymbol{U}_{z_t}^d)^2, \tag{5.12}$$

where $\mathrm{dist}(\boldsymbol{\xi}_t, \boldsymbol{\mu}_{z_t}, \boldsymbol{U}_{z_t}^d)^2$ represents the distance of the datapoint $\boldsymbol{\xi}_t$ to the subspace of cluster z_t defined by mean $\boldsymbol{\mu}_{z_t}$ and unit eigenvectors of the covariance matrix $\boldsymbol{U}_{z_t}^d$ (see Eq. (5.13) below), and λ, λ_1 represent the penalty terms for the number of clusters and the subspace dimension of each cluster respectively. The algorithm optimizes the number of clusters and the subspace dimension of each cluster while minimizing the distance of the datapoints to the respective subspaces of each cluster. Note that the clustering objective is similar to the DP-means algorithm except that the distance to the cluster means is replaced by the distance to the subspace of the cluster and an added penalty is placed on choosing clusters with more subspace dimensions. In other words, DP-GMM is the limiting case of DP-MPPCA with very large penalty on the subspace dimension.

[3] Note that MPPCA is closely related to MFA, and uses isotropic noise matrix instead of the diagonal noise matrix used in MFA as we have seen in the previous chapter.

5.4.2 Online Inference in DP-MPPCA

In the online setting, we seek to incrementally update the parameters $\Theta_{t,\text{MPPCA}}$ (Θ_{MPPCA} at time t) with the new observation $\boldsymbol{\xi}_{t+1}$ without having to retrain the model in a batch manner and store the demonstration data. The parameters are updated in two steps: the cluster assignment step followed by the parameter updates step.

5.4.2.1 Cluster Assignment z_{t+1}:

The cluster assignment z_{t+1} of $\boldsymbol{\xi}_{t+1}$ in the online case follows the same principle as in Eq. (5.7), except the distance is now computed from the subspace of a cluster $\text{dist}(\boldsymbol{\xi}_{t+1}, \boldsymbol{\mu}_{t,i}, \boldsymbol{U}_{t,i}^{d_{t,i}})^2$, defined using the difference between the mean-centered datapoint and the mean-centered datapoint projected upon the subspace $\boldsymbol{U}_{t,i}^{d_{t,i}} \in \mathbb{R}^{D \times d_{t,i}}$ spanned by the $d_{t,i}$ unit eigenvectors of the covariance matrix, i.e.,

$$\text{dist}(\boldsymbol{\xi}_{t+1}, \boldsymbol{\mu}_{t,i}, \boldsymbol{U}_{t,i}^{d_{t,i}}) = \left\| (\boldsymbol{\xi}_{t+1} - \boldsymbol{\mu}_{t,i}) - \rho_i \boldsymbol{U}_{t,i}^{d_{t,i}} \boldsymbol{U}_{t,i}^{d_{t,i}\top} (\boldsymbol{\xi}_{t+1} - \boldsymbol{\mu}_{t,i}) \right\|_2, \tag{5.13}$$

where

$$\rho_i = \exp\left(-\frac{\|\boldsymbol{\xi}_{t+1} - \boldsymbol{\mu}_{t,i}\|_2^2}{b_m} \right)$$

weighs the projected mean-centered datapoint according to the distance of the datapoint from the cluster center ($0 < \rho_i \leq 1$). Its effect is controlled by the bandwidth parameter b_m. If b_m is large, then the far away clusters have a greater influence; otherwise nearby clusters are favored. Note that ρ_i assigns more weight to the projected mean-centered datapoint for the nearby clusters than the distant clusters to limit the size of the cluster/subspace. Our subspace distance formulation is different from [Wang 2015] as we weigh the subspace of the nearby clusters more than the distant clusters. This allows us to avoid clustering all the datapoints in the same subspace (near or far) together. The cluster assignment is deterministically updated using

$$z_{t+1} = \underset{i=1\ldots K+1}{\arg\min} \begin{cases} \text{dist}(\boldsymbol{\xi}_{t+1}, \boldsymbol{\mu}_{t,i}, \boldsymbol{U}_{t,i}^{d_{t,i}})^2, & \text{if } i \leq K \\ \lambda, & \text{otherwise.} \end{cases} \tag{5.14}$$

5.4.2.2 Parameter Updates $\Theta_{t+1,\text{MPPCA}}$:

Given the cluster assignment $z_{t+1} = i$ at time $t+1$, the prior and the mean of the assigned cluster are updated in the same way as DP-GMM (see Eq. (5.8)). Depending upon the

nature of the streaming data, $w_{t+1,i}$ can be updated as follows:

- For stationary online learning problems where the data is sampled from some fixed distribution, we update the weight $w_{t+1,i}$ linearly with the number of instances belonging to that cluster, namely

$$w_{t+1,i} = w_{t,i} + 1, \qquad w_{0,i} = 1. \tag{5.15}$$

- For non-stationary online learning problems where the distribution of streaming data varies over time, we update the weight vector based on the *eligibility trace* that takes into account the temporary occurrence of visiting a particular cluster.[4] The trace indicates how much a cluster is eligible for undergoing changes with the new parameter update. The trace is updated such that the weights of all the clusters are decreased by the discount factor $\zeta \in (0, 1)$ and the weight of the visited cluster is incremented, i.e., the more often a state is visited, the higher is the eligibility weight of all the previous updates relative to the new parameter update, namely

$$w_{t+1,i} = \begin{cases} \zeta w_{t,i} + 1, & \text{if } i = z_{t+1} \\ \zeta w_{t,i}, & \text{if } i \neq z_{t+1}. \end{cases} \tag{5.16}$$

- For non-stationary problems where learning is continuous and may not depend upon the number of datapoints, the weight vector is kept constant $w_{t+1,i} = w_{t,i} = w^*$ at all time steps as a step-size parameter.

The covariance matrix could then be updated online as

$$\bar{\boldsymbol{\Sigma}}_{t+1,i} = \frac{w_{t,i}}{w_{t,i}+1}\, \boldsymbol{\Sigma}_{t,i} + \frac{w_{t,i}}{(w_{t,i}+1)^2}\, (\boldsymbol{\xi}_{t+1} - \boldsymbol{\mu}_{t+1,i})(\boldsymbol{\xi}_{t+1} - \boldsymbol{\mu}_{t+1,i})^\top. \tag{5.17}$$

However, updating the covariance matrix online in D-dimensional space can be prohibitively expensive for even moderate size problems. To update the covariance matrix in its intrinsic lower dimension, similarly to [Bellas 2013], we compute $\boldsymbol{g}_{t+1,i} \in \mathbb{R}^{d_i}$ as the projection of datapoint $\boldsymbol{\xi}_{t+1}$ onto the existing set of basis vectors of $\boldsymbol{U}_{t,i}^{d_{t,i}}$. Note that the cardinality of basis vectors is different for each covariance matrix. If the datapoint belongs to the subspace of $\boldsymbol{U}_{t,i}^{d_{t,i}}$, the retro-projection of the datapoint in its original space, as given by the residual vector $\boldsymbol{p}_{t+1,i} \in \mathbb{R}^D$, would be a zero vector; otherwise the residual vector belongs to the null space of $\boldsymbol{U}_{t,i}^{d_{t,i}}$, and its unit vector $\tilde{\boldsymbol{p}}_{t+1,i}$ needs to be added to the

[4] Eligibility traces are commonly used in reinforcement learning to evaluate the state for undergoing learning changes in temporal-difference learning [Sutton 1998].

existing set of basis vectors, i.e.,

$$\begin{aligned}
\boldsymbol{g}_{t+1,i} &= \boldsymbol{U}_{t,i}^{d_{t,i}\top}(\boldsymbol{\xi}_{t+1} - \boldsymbol{\mu}_{t,i}),\\
\boldsymbol{p}_{t+1,i} &= (\boldsymbol{\xi}_{t+1} - \boldsymbol{\mu}_{t,i}) - \boldsymbol{U}_{t,i}^{d_{t,i}}\, \boldsymbol{g}_{t+1,i},\\
\tilde{\boldsymbol{p}}_{t+1,i} &= \begin{cases} \frac{\boldsymbol{p}_{t+1,i}}{\|\boldsymbol{p}_{t+1,i}\|_2}, & \text{if } \|\boldsymbol{p}_{t+1,i}\|_2 > 0\\ \boldsymbol{0}_D, & \text{otherwise.}\end{cases}
\end{aligned}$$

The new set of basis vectors augmented with the unit residual vector is represented as

$$\boldsymbol{U}_{t+1,i}^{d_{t,i}} = [\boldsymbol{U}_{t,i}^{d_{t,i}}\ ,\ \tilde{\boldsymbol{p}}_{t+1,i}]\ \boldsymbol{R}_{t+1,i}, \tag{5.18}$$

where $\boldsymbol{R}_{t+1,i} \in \mathbb{R}^{(d_{t,i}+1)\times(d_{t,i}+1)}$ is the rotation matrix to incrementally update the augmented basis vectors. $\boldsymbol{R}_{t+1,i}$ is obtained by simplifying the eigendecomposition problem

$$\bar{\boldsymbol{\Sigma}}_{t+1,i} = \boldsymbol{U}_{t+1,i}^{d_{t,i}}\ \boldsymbol{\Sigma}_{t+1,i}^{(\text{diag})}\ \boldsymbol{U}_{t+1,i}^{d_{t,i}\ \top}. \tag{5.19}$$

Substituting the value of $\bar{\boldsymbol{\Sigma}}_{t+1,i}$ from Eq. (5.17) and $\boldsymbol{U}_{t+1,i}^{d_{t,i}}$ from (5.18) yields the reduced eigendecomposition problem of size $(d_{t,i}+1)\times(d_{t,i}+1)$ with

$$\frac{w_{t,i}}{w_{t,i}+1}\begin{bmatrix}\boldsymbol{\Sigma}_{t,i}^{(\text{diag})} & \boldsymbol{0}_{d_{t,i}}\\ \boldsymbol{0}_{d_{t,i}}^\top & 0\end{bmatrix} + \frac{w_{t,i}}{(w_{t,i}+1)^2}\begin{bmatrix}\boldsymbol{g}_{t+1,i}\,\boldsymbol{g}_{t+1,i}^\top & \nu_i\boldsymbol{g}_{t+1,i}\\ \nu_i\boldsymbol{g}_{t+1,i}^\top & \nu_i^2\end{bmatrix} = \boldsymbol{R}_{t+1,i}\ \boldsymbol{\Sigma}_{t+1,i}^{(\text{diag})}\ \boldsymbol{R}_{t+1,i}^\top, \tag{5.20}$$

where $\nu_i = \tilde{\boldsymbol{p}}_{t+1,i}^\top(\boldsymbol{\xi}_{t+1} - \boldsymbol{\mu}_{t+1,i})$. Solving for $\boldsymbol{R}_{t+1,i}$ and substituting it in Eq. (5.18) gives the required updates of the basis vectors in a computationally and memory efficient manner. The subspace dimension of the i-th mixture component is updated by keeping an estimate of the average distance vector $\bar{\boldsymbol{e}}_{t,i} \in \mathbb{R}^D$ whose k-th element represents the mean distance of the datapoints to the $(k-1)$ subspace basis vectors of $\boldsymbol{U}_{t,i}^k$ for the i-th cluster. Let us denote $\boldsymbol{\delta}_i$ as the vector measuring the distance of the datapoint $\boldsymbol{\xi}_{t+1}$ to each of the subspaces of $\boldsymbol{U}_{t,i}^k$ for the i-th cluster where $k = \{0\ldots(d_{t,i}+1)\}$, i.e.,

$$\boldsymbol{\delta}_i = \begin{bmatrix}\text{dist}(\boldsymbol{\xi}_{t+1}, \boldsymbol{\mu}_{t+1,i}, \boldsymbol{U}_{t+1,i}^0)^2\\ \vdots\\ \text{dist}(\boldsymbol{\xi}_{t+1}, \boldsymbol{\mu}_{t+1,i}, \boldsymbol{U}_{t+1,i}^{d_{t,i}+1})^2\end{bmatrix}, \tag{5.21}$$

where $\text{dist}(\boldsymbol{\xi}_{t+1}, \boldsymbol{\mu}_{t+1,i}, \boldsymbol{U}_{t+1,i}^0)^2$ is the distance to the cluster subspace with 0 dimension (the cluster center point), $\text{dist}(\boldsymbol{\xi}_{t+1}, \boldsymbol{\mu}_{t+1,i}, \boldsymbol{U}_{t+1,i}^1)^2$ is the distance to the cluster subspace with 1 dimension (the line), and so on. The average distance vector $\bar{\boldsymbol{e}}_{t+1,i}$ and the subspace

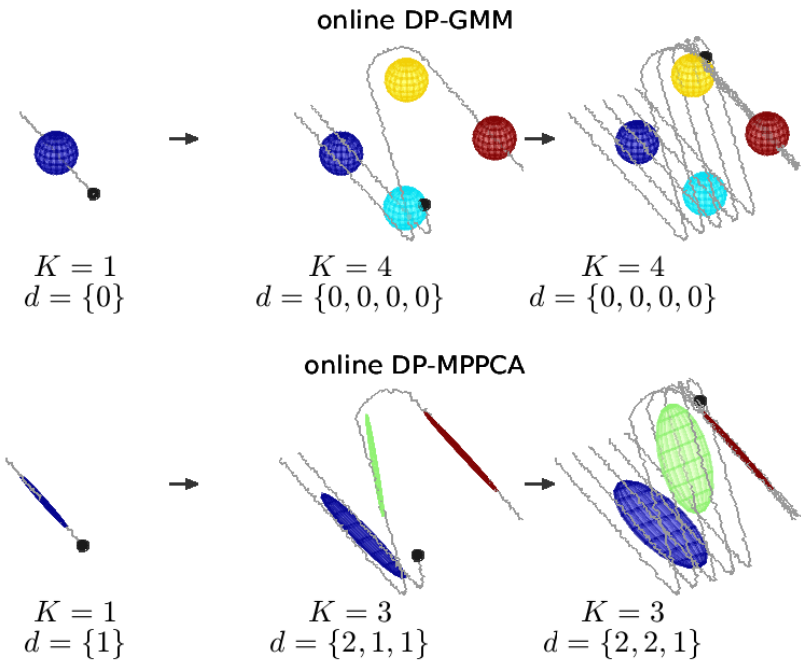


Figure 5.3: Non-parametric online clustering of Z-shaped streaming data under small variance asymptotics with: *(top)* online DP-GMM, *(bottom)* online DP-MPPCA.

dimension $d_{t+1,i}$ are incrementally updated as

$$\bar{\boldsymbol{e}}_{t+1,i} = \frac{1}{w_{t,i} + 1}\Big(w_{t,i}\bar{\boldsymbol{e}}_{t,i} + \boldsymbol{\delta}_i\Big), \tag{5.22}$$

$$d_{t+1,i} = \underset{d=0:D-1}{\arg\min}\Big\{\lambda_1 d + \bar{\boldsymbol{e}}_{t+1,i}\Big\}. \tag{5.23}$$

Given the updated set of basis vectors, the projection matrix and the covariance matrix are updated as

$$\boldsymbol{\Lambda}_{t+1,i}^{d_{t+1,i}} = \boldsymbol{U}_{t+1,i}^{d_{t+1,i}} \sqrt{\boldsymbol{\Sigma}_{t+1,i}^{(\text{diag})}}, \tag{5.24}$$

$$\boldsymbol{\Sigma}_{t+1,i} = \boldsymbol{\Lambda}_{t+1,i}^{d_{t+1,i}} \boldsymbol{\Lambda}_{t+1,i}^{d_{t+1,i}\top} + \sigma^2 \boldsymbol{I}. \tag{5.25}$$

Loss function $\mathcal{L}(z_{t+1}, d_{t+1,z_{t+1}}, \boldsymbol{\mu}_{t+1,z_{t+1}}, \boldsymbol{U}_{t+1,z_{t+1}}^{d_{t+1,z_{t+1}}})$: The loss function optimized at time step $t+1$ is

$$\begin{aligned}\mathcal{L}(z_{t+1}, d_{t+1,z_{t+1}}, \boldsymbol{\mu}_{t+1,z_{t+1}}, \boldsymbol{U}_{t+1,z_{t+1}}^{d_{t+1,z_{t+1}}}) = \lambda K + \lambda_1 d_{t+1,z_{t+1}} + \text{dist}(\boldsymbol{\xi}_{t+1}, \boldsymbol{\mu}_{t+1,z_{t+1}}, \boldsymbol{U}_{t+1,z_{t+1}}^{d_{t+1,z_{t+1}}})^2 \\ \leq \mathcal{L}(z_{t+1}, d_{t,z_{t+1}}, \boldsymbol{\mu}_{t,z_{t+1}}, \boldsymbol{U}_{t,z_{t+1}}^{d_{t,z_{t+1}}}).\end{aligned}$$

The loss function provides an intuitive trade-off between the fitness term $\text{dist}(\boldsymbol{\xi}_{t+1}, \boldsymbol{\mu}_{t+1,z_{t+1}}, \boldsymbol{U}_{t+1,z_{t+1}}^{d_{t+1,z_{t+1}}})^2$ and the model selection parameters K and d_k. Increasing the number of clusters or the subspace dimension of the assigned cluster decreases the distance of the datapoint to the assigned subspace at the cost of penalty terms λ and λ_1. Parameters of the assigned cluster are updated in a greedy manner such that the loss function is guaranteed to decrease at the current time step. In case a new cluster is assigned to the datapoint, the loss function at time t is evaluated with the cluster having the lowest cost among the existing set of clusters. Note that setting $d_{t,i} = 0$ by choosing $\lambda_1 \gg 0$ gives the same loss function and objective function as the online DP-GMM algorithm with isotropic Gaussians.

To illustrate the difference of encoding between online DP-means and online DP-MPPCA, we evaluate the performance of the algorithms on a Z-shaped 3-dimensional stream of datapoints with penalty parameters $\{\lambda = 35, \sigma^2 = 100\}$ for online DP-GMM, and $\{\lambda = 14, \lambda_1 = 2, \sigma^2 = 1, b_m = 1 \times 10^4\}$ for online DP-MPPCA. Fig. 5.3 shows that online DP-GMM under small variance asymptotics fails to represent the variance in the demonstrations with $d = 0$, whereas the number of clusters and the subspace dimension adequately evolves for online DP-MPPCA to model the underlying distribution.

5.5 Online HDP-HSMM

We now consider the Bayesian non-parametric extension of HSMM and present our incremental formulation to estimate the parameters of an infinite HSMM, , $\Theta_{\text{HSMM}} = \left\{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \{a_{i,m}\}_{m=1}^K, \mu_i^S, \Sigma_i^S\right\}_{i=1}^K$, where the output distribution of i-th state is represented by a parsimonious multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Lambda}_i^{d_i}\, \boldsymbol{\Lambda}_i^{d_i\top} + \sigma^2 \boldsymbol{I})$. Compared to the previous section, transition probabilities and an explicit state duration model for each state are introduced as additional parameters.

5.5.1 Hierarchical Dirichlet Process HSMM (HDP-HSMM)

Specifying the number of latent states in an HMM/HSMM is often difficult. Model selection methods such as cross-validation or Bayesian Information Criterion (BIC) are typically used to determine the number of states. Bayesian non-parametric approaches comprising of HDPs provide a principled model selection procedure by Bayesian inference in an HMM/HSMM with an infinite number of states. Interested readers can find details of DPs and HDPs for specifying an infinite set of conditional transition distribution priors in

[Teh 2006].

HDP-HMM [Beal 2002, Van Gael 2008] is an infinite state Bayesian non-parametric generalization of the HMM with HDP prior on the transition distribution. In this model, the state transition distribution for each state follows a Dirichlet process $\boldsymbol{G}_i \sim \mathrm{DP}(\alpha^{\mathrm{DP}}, \boldsymbol{G}_0)$ with concentration parameter α^{DP} and shared base distribution $\boldsymbol{G}_0$, such that $\boldsymbol{G}_0$ is the global Dirichlet process $\boldsymbol{G}_0 \sim \mathrm{DP}(\gamma^{\mathrm{DP}}, \boldsymbol{H})$ with concentration parameter γ^{DP} and base distribution $\boldsymbol{H}$. The top level DP enables sharing of the existing states with a new state created under a bottom level DP for each state and encourages visiting of the same consistent set of states in the sequence. Let $\boldsymbol{\beta}$ denote the weights of $\boldsymbol{G}_0$ in its stick-breaking construction [Sethuraman 1994], then the non-parametric approach takes the form

$$\begin{aligned}
\boldsymbol{\beta}|\gamma^{\mathrm{DP}} &\sim \mathrm{GEM}(\gamma^{\mathrm{DP}}), \\
\boldsymbol{\pi}_i|\alpha^{\mathrm{DP}}, \boldsymbol{\beta} &\sim \mathrm{DP}(\alpha^{\mathrm{DP}}, \boldsymbol{\beta}), \\
\{\boldsymbol{\mu}_i, \boldsymbol{\Lambda}_i^{d_i}, d_i\} &\sim \boldsymbol{H}, \\
z_t &\sim \mathrm{Mult}(\boldsymbol{\pi}_{z_{t-1}}), \\
\boldsymbol{\xi}_t|z_t &\sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Lambda}_i^{d_i}\boldsymbol{\Lambda}_i^{d_i\top} + \sigma^2\boldsymbol{I}),
\end{aligned}$$

where GEM represents the Griffiths, Engen and McCloskey distribution [Pitman 2002], and we have used the parsimonious representation of a Gaussian for the output distribution of a state without loss of generality.

Johnson et al. presented an extension of HDP-HMM to HDP-HSMM by explicitly drawing the state duration distribution parameters and precluding the self-transitions [Johnson 2013]. Other extensions such as sticky HDP-HMM [Fox 2008] add a self-transition bias parameter to the DP of each state to prolong the state-dwell times. We take a simpler approach to explicitly encode the state duration by setting the self-transition probabilities to zero and estimating the parameters $\{\mu_i^S, \Sigma_i^S\}$ empirically from the hidden state sequence $\{z_1, \ldots, z_T\}$.

Note that learning the model in this Bayesian non-parametric setting involves computing the posterior distribution over the latent state, the output state distribution and the transition distribution parameters. The problem is more challenging than the maximum likelihood parameter estimation of HMMs and requires MCMC sampling or variational inference techniques to compute the posterior distribution. Performing small-variance asymptotics of the joint likelihood of HDP-HMM, on the other hand, yields the maximum aposteriori

estimates of the parameters by iteratively minimizing the loss function[5]

$$\mathcal{L}(\boldsymbol{z}, \boldsymbol{d}, \boldsymbol{\mu}, \boldsymbol{U}, \boldsymbol{a}) = \sum_{t=1}^{T} \text{dist}(\boldsymbol{\xi}_t, \boldsymbol{\mu}_{z_t}, \boldsymbol{U}_{z_t}^{d_i})^2 + \lambda(K-1) + \lambda_1 \sum_{i=1}^{K} d_i - \lambda_2 \sum_{t=1}^{T-1} \log(a_{z_t, z_{t+1}}) + \lambda_3 \sum_{i=1}^{K} (\tau_i - 1),$$

where $\lambda_2, \lambda_3 > 0$ are the additional penalty terms responsible for prolonging the state duration estimates compared to the loss function in Eq. (5.12). The λ_2 term favours the transitions to states with higher transition probability (states which have been visited more often before), λ_3 penalizes for transition to unvisited states with τ_i denoting the number of distinct transitions out of state i, and λ, λ_1 are the penalty terms for increasing the number of states and the subspace dimension of each output state distribution.

5.5.2 Online Inference in HDP-HSMM

For the online setting, we denote the parameter set Θ_{HSMM} at time t as $\Theta_{t,\text{HSMM}}$. Given the observation $\boldsymbol{\xi}_{t+1}$, we now present the cluster assignment and the parameter update steps for the online incremental version of HDP-HSMM.

5.5.2.1 Cluster Assignment z_{t+1}:

The datapoint $\boldsymbol{\xi}_{t+1}$ is assigned to cluster z_{t+1} based on the rule

$$z_{t+1} = \underset{i=1:K+1}{\arg\min} \begin{cases} \text{dist}(\boldsymbol{\xi}_{t+1}, \boldsymbol{\mu}_{t,i}, \boldsymbol{U}_{t,i}^{d_{t,i}})^2 - \lambda_2 \log a_{t,z_t,i}, & \text{if } \{a_{t,z_t,i} > 0, i \leq K\} \quad (5.26) \\ \text{dist}(\boldsymbol{\xi}_{t+1}, \boldsymbol{\mu}_{t,i}, \boldsymbol{U}_{t,i}^{d_{t,i}})^2 - \lambda_2 \log \frac{1}{\sum_{k=1}^{K} c_{t,z_t,k} + 1} + \lambda_3, & \text{if } \{a_{t,z_t,i} = 0, i \leq K\} \quad (5.27) \\ \lambda - \lambda_2 \log \frac{1}{\sum_{k=1}^{K} c_{t,z_t,k} + 1} + \lambda_3, & \text{otherwise,} \quad (5.28) \end{cases}$$

where $c_{t,i,j}$ is an auxiliary transition variable that counts the number of visits from state i to state j till time t. The assignment procedure evaluates the cost on two main criteria: 1) distance of the datapoint to the existing cluster subspaces given by $\text{dist}(\boldsymbol{\xi}_{t+1}, \boldsymbol{\mu}_{t,i}, \boldsymbol{U}_{t,i}^{d_i})$, and 2) transition probability of moving from the current state to the state $a_{t,z_t,i}$. The

[5]Setting $d_i = 0$ by choosing $\lambda_1 \gg 0$ gives the loss function formulation with isotropic Gaussian under small variance asymptotics as shown in [Roychowdhury 2013].

procedure favours the next state to be one whose distance from the subspace of a cluster is low and whose transition probability is high, as seen in Eq. (5.26). If the probability of transitioning to a given state is zero, an additional penalty of λ_3 is added along with a pseudo transition count to that state $\frac{1}{\sum_{k=1}^{K} c_{t,z_t,k}+1}$. Finally, if the cost of transitioning to a new state at subspace distance λ in Eq. (5.28) is lower than the cost evaluated in Eq. (5.26) and Eq. (5.27), a new cluster is created with the datapoint and default parameters.

5.5.2.2 Parameter Updates $\Theta_{t+1,\mathbf{HSMM}}$:

Given the cluster assignment $z_{t+1} = i$, we first estimate the parameters $\boldsymbol{\mu}_{t+1,i}, \boldsymbol{U}_{t+1,i}^{d_{t,i}}, d_{t+1,i}$, and $\boldsymbol{\Sigma}_{t+1,i}$ following the update rules in Eqs (5.8), (5.18), (5.23) and (5.25), respectively. We update the transition probabilities via the auxiliary transition count matrix with

$$c_{t+1,z_t,z_{t+1}} = c_{t,z_t,z_{t+1}} + 1, \tag{5.29}$$

$$a_{t+1,z_t,z_{t+1}} = c_{t+1,z_t,z_{t+1}} \,/\, \sum_{k=1}^{K} c_{t+1,z_t,k}. \tag{5.30}$$

To update the state duration probabilities, we keep a count of the duration steps s_t in which the cluster assignment is the same, i.e.,

$$s_{t+1} = \begin{cases} s_t + 1, & \text{if } z_{t+1} = z_t, \\ 0, & \text{otherwise.} \end{cases} \tag{5.31}$$

Let us denote n_{t,z_t} as the total number of transitions to other states from the state z_t till time t. When the subsequent cluster assignment is different, $z_{t+1} \neq z_t$, the duration count is reset to zero, $s_{t+1} = 0$, the transition count to other states is incremented, $n_{t+1,z_t} = n_{t,z_t}+1$, and the duration model parameters $\{\mu_{t+1,z_t}^S, \Sigma_{t+1,z_t}^S\}$ are updated as

$$\mu_{t+1,z_t}^S = \mu_{t,z_t}^S + \frac{(s_t - \mu_{t,z_t}^S)}{n_{t,z_t} + 1}, \tag{5.32}$$

$$e_{t+1,z_t} = e_{t,z_t} + (s_t - \mu_{t,z_t}^S)(s_t - \mu_{t+1,z_t}^S), \tag{5.33}$$

$$\Sigma_{t+1,z_t}^S = \frac{e_{t+1,z_t}}{n_{t,z_t}}. \tag{5.34}$$

Loss function $\mathcal{L}(z_{t+1}, d_{t+1,z_{t+1}}, \boldsymbol{\mu}_{t+1,z_{t+1}}, \boldsymbol{U}_{t+1,z_{t+1}}^{d_{t+1,z_{t+1}}}, a_{t+1,z_t,z_{t+1}})$: The parameters updated

at time step $t+1$ minimizes the loss function

$$\mathcal{L}(z_{t+1}, d_{t+1,z_{t+1}}, \boldsymbol{\mu}_{t+1,z_{t+1}}, \boldsymbol{U}_{t+1,z_{t+1}}^{d_{t+1,z_{t+1}}}, a_{t+1,z_t,z_{t+1}}) = \lambda(K-1) + \lambda_1 d_{t+1,z_{t+1}} - \lambda_2 \log(a_{z_t,z_{t+1}}) + \lambda_3\, \tau_{z_{t+1}} + \mathrm{dist}(\boldsymbol{\xi}_{t+1}, \boldsymbol{\mu}_{t+1,z_{t+1}}, \boldsymbol{U}_{t+1,z_{t+1}}^{d_{t+1,z_{t+1}}})^2 \leq \mathcal{L}(z_{t+1}, d_{t,z_{t+1}}, \boldsymbol{\mu}_{t,z_{t+1}}, \boldsymbol{U}_{t,z_{t+1}}^{d_{t,z_{t+1}}}, a_{t,z_t,z_{t+1}}). \quad (5.35)$$

A decrease of the loss function ensures that the assigned cluster parameters are updated in an optimal manner. In case a new cluster is assigned to the datapoint, the loss function at time t is evaluated with the cluster having the lowest cost among the existing set of clusters.

Remark: Note that λ_2 encourages visiting the more influential states, and λ_3 restricts creation of new states. We do not explicitly penalize the deviation from the state duration distribution in the cluster assignment step or the loss function, and only re-estimate the parameters of the state duration in the parameter update step. Deviation from the state duration parameters may also be explicitly penalized as shown with small variance asymptotic analysis of hidden Markov jump processes [Huggins 2015].

5.6 Scalable Online Sequence Clustering (SOSC)

SOSC is an unsupervised non-parametric online learning algorithm for clustering time-series data. It incrementally projects the streaming data in its low dimensional subspace and maintains a history of the duration steps and the subsequent transition to other subspaces. The projection mechanism uses a non-parametric locally linear principal component analysis whose redundant dimensions are automatically discarded by small variance asymptotic analysis along those dimensions, while the spatio-temporal information is stored with an infinite state hidden semi-Markov model. During learning, if a cluster evolves such that it is closer to another cluster than the threshold λ, the two clusters are merged into one and the subspace of the dominant cluster is retained. The overall algorithm is shown in Alg. 5.

The algorithm yields a generative model that scales well in higher dimensions and does not require computation of numerically unstable gradients for the parameter updates at each iteration. These desirable aspects of the model comes at a cost of hard/deterministic clusters which could be a bottleneck for some applications. Non-parametric treatment aids the user to build the model online without specifying the number of clusters and the sub-

Algorithm 5 *Scalable Online Sequence Clustering (SOSC)*

Input: $< \lambda, \lambda_1, \lambda_2, \lambda_3, \sigma^2, b_m >$
procedure SOSC
Initialize $K := 1, \{d_{0,K},\ c_{0,K,K},\ \mu^S_{0,K},\ n_{0,K},\ e_K\} := 0$
while new $\boldsymbol{\xi}_{t+1}$ is added **do**
  Assign cluster z_{t+1} to $\boldsymbol{\xi}_{t+1}$ using cases in Eq. (5.26), Eq. (5.27) and Eq. (5.28)
  if $z_{t+1} = K + 1$ **then**
    $K := K + 1, \quad \boldsymbol{\mu}_{t+1,K} := \boldsymbol{\xi}_t, \quad \boldsymbol{\Sigma}_{t+1,K} := \sigma^2 \boldsymbol{I}$
    $\{d_{t+1,K},\ c_{t+1,K,K},\ \mu^S_{old,K},\ n_{t+1,K},\ e_{t+1,K}\} := 0$
  else
    Update $\boldsymbol{\mu}_{t+1,z_{t+1}}$ using Eq. (5.8)
    Solve $\boldsymbol{R}_{t+1,z_{t+1}}$, update $\boldsymbol{U}^{d_{t,z_{t+1}}}_{t+1,z_{t+1}}$ using Eq. (5.18)
    Update $d_{t+1,z_{t+1}}$ using Eq. (5.23)
    Update $\boldsymbol{\Sigma}_{t+1,z_{t+1}}$ using Eq. (5.25)
  end if
  Update $c_{t+1,z_t,z_{t+1}}, a_{t+1,z_t,z_{t+1}}$ using Eq. (5.29), (5.30)
  if $z_{t+1} = z_t$ **then**
    $s_{t+1} := s_t + 1$
  else
    $s_{t+1} := 0, \qquad n_{t+1,z_t} := n_{t,z_t} + 1$
    Update μ^S_{t+1,z_t} using Eq. (5.32)
    Update e_{t+1,z_t} using Eq. (5.33)
    Update Σ^S_{t+1,z_t} using Eq. (5.34) for $n_{t,z_t} > 1$
  end if
  $z_t := z_{t+1}$
  for $i := 1$ **to** K **do**
    if $\|\boldsymbol{\mu}_{t+1,z_{t+1}} - \boldsymbol{\mu}_{t,i}\|_2 < \lambda$ **then** $\{i \neq z_{t+1}\}$
      Merge_Clusters(z_{t+1}, i)
    end if
  end for
end while
return $\theta^*_{t,s} = \{\boldsymbol{\mu}_{t,i}, \boldsymbol{\Sigma}_{t,i}, \{a_{t,i,j}\}^K_{j=1}, \mu^S_{t,i}, \Sigma^S_{t,i}\}^K_{i=1}$

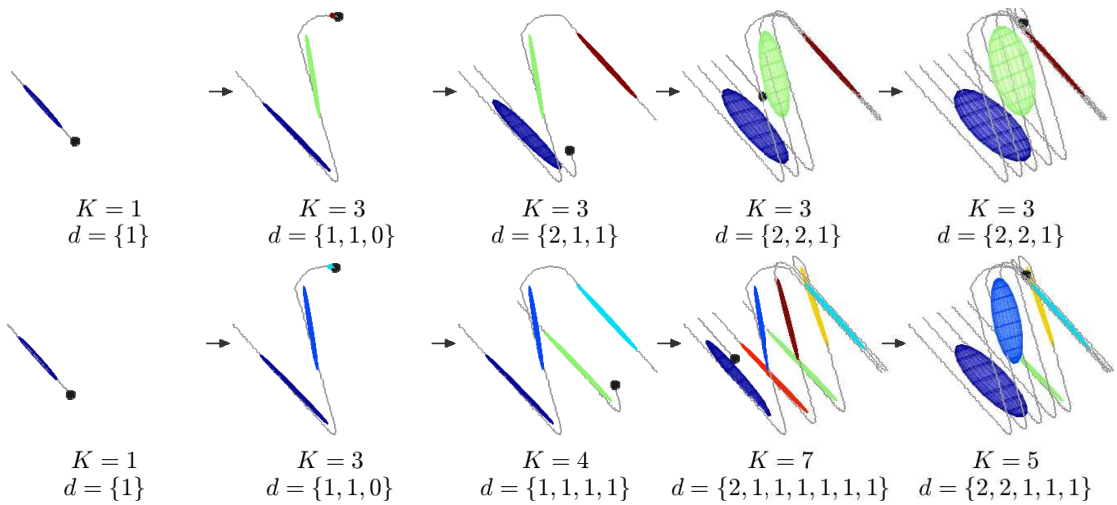


Figure 5.4: Non-parametric online clustering of Z-shaped streaming data under small variance asymptotics with different ordering of data on top and bottom can lead to different models.

space dimension of each cluster, as the parameter set grows with the size/complexity of the data during learning. The penalty parameters introduced are more intuitive to specify and act as regularization terms for model selection based on the structure of the data. Note that the order of the streaming data plays an important role during learning, and multiple starts from different initial configurations may lead to different solutions as we update the model parameters after registering every new sample (see Fig. 5.4). Alternatively, the model parameters can be initialized with a batch algorithm after storing a few demonstrations, or the parameters can be updated sequentially in a mini-batch manner. Systematic investigation of these approaches is subject to future work.

5.6.1 Task-Parameterized Formulation of SOSC

Task-parameterized models provide a probabilistic formulation to deal with different real world situations by adapting the model parameters in accordance with the external task parameters that describe the environment/configuration/situation, instead of hard coding the solution for each new situation or handling it in an *ad hoc* manner [Wilson 1999, Tanwani 2016a]. When a different situation occurs (position/orientation of the object changes), changes in the task parameters/reference frames are used to modulate the model parameters in order to adapt the robot movement to the new situation.

For the online setting, we represent the task parameters with P coordinate systems, defined by $\{\boldsymbol{A}_{t,j}, \boldsymbol{b}_{t,j}\}_{j=1}^{P}$, where $\boldsymbol{A}_{t,j}$ denotes the orientation of the frame as a rotation matrix and

$\boldsymbol{b}_{t,j}$ represents the origin of the frame at time t. Each datapoint $\boldsymbol{\xi}_t$ is observed from the viewpoint of P different experts/frames, with $\boldsymbol{\xi}_t^{(j)} = \boldsymbol{A}_{t,j}^{-1}(\boldsymbol{\xi}_t - \boldsymbol{b}_{t,j})$ denoting the datapoint observed with respect to frame j. The parameters of the task-parameterized SOSC model are defined by $\theta_{t,\text{TP-HSMM}} = \left\{\{\boldsymbol{\mu}_{t,i}^{(j)}, \boldsymbol{\Sigma}_{t,i}^{(j)}\}_{j=1}^{P}, \{a_{t,i,m}\}_{m=1}^{K}, \mu_{t,i}^{S}, \Sigma_{t,i}^{S}\right\}_{i=1}^{K}$, where $\boldsymbol{\mu}_{t,i}^{(j)}$ and $\boldsymbol{\Sigma}_{t,i}^{(j)}$ define the mean and the covariance matrix of i-th mixture component in frame j at time t. Parameter updates of the task-parameterized SOSC algorithm remain the same as described in Alg. 5, except the computation of the mean and the covariance matrix is repeated for each coordinate system separately.

In order to combine the output of the experts for an unseen situation represented by the frames $\{\tilde{\boldsymbol{A}}_{t,j}, \tilde{\boldsymbol{b}}_{t,j}\}_{j=1}^{P}$, we linearly transform the Gaussians back to the global coordinates with $\{\tilde{\boldsymbol{A}}_{t,j}, \tilde{\boldsymbol{b}}_{t,j}\}_{j=1}^{P}$, and retrieve the new model parameters $\{\tilde{\boldsymbol{\mu}}_{t,i}, \tilde{\boldsymbol{\Sigma}}_{t,i}\}$ for the i-th mixture component by computing the products of the linearly transformed Gaussians

$$\mathcal{N}(\tilde{\boldsymbol{\mu}}_{t,i}, \tilde{\boldsymbol{\Sigma}}_{t,i}) \propto \prod_{j=1}^{P} \mathcal{N}\left(\tilde{\boldsymbol{A}}_{t,j}\boldsymbol{\mu}_{t,i}^{(j)} + \tilde{\boldsymbol{b}}_{t,j}, \tilde{\boldsymbol{A}}_{t,j}\boldsymbol{\Sigma}_{t,i}^{(j)}\tilde{\boldsymbol{A}}_{t,j}^{\top}\right). \tag{5.36}$$

The product of Gaussians can be evaluated in a closed form solution with

$$\begin{aligned}
\tilde{\boldsymbol{\mu}}_{t,i} &= \tilde{\boldsymbol{\Sigma}}_{t,i} \sum_{j=1}^{P} \left(\tilde{\boldsymbol{A}}_{t,j}\boldsymbol{\Sigma}_{t,i}^{(j)}\tilde{\boldsymbol{A}}_{t,j}^{\top}\right)^{-1} \left(\tilde{\boldsymbol{A}}_{t,j}\boldsymbol{\mu}_{t,i}^{(j)} + \tilde{\boldsymbol{b}}_{t,j}\right), \\
\tilde{\boldsymbol{\Sigma}}_{t,i} &= \left(\sum_{j=1}^{P} \left(\tilde{\boldsymbol{A}}_{t,j}\boldsymbol{\Sigma}_{t,i}^{(j)}\tilde{\boldsymbol{A}}_{t,j}^{\top}\right)^{-1}\right)^{-1}.
\end{aligned} \tag{5.37}$$

Under the small variance asymptotics, the loss function at time step $t+1$ for the task-parametrized SOSC model with the resulting $\mathcal{N}(\tilde{\boldsymbol{\mu}}_{t+1,z_{t+1}}, \tilde{\boldsymbol{\Sigma}}_{t+1,z_{t+1}})$ yields

$$\begin{aligned}
\mathcal{L}(z_{t+1}, \tilde{d}_{t+1,z_{t+1}}, \tilde{\boldsymbol{\mu}}_{t+1,z_{t+1}}, \tilde{\boldsymbol{U}}_{t+1,z_{t+1}}^{\tilde{d}_{t+1,z_{t+1}}}, a_{t+1,z_t,z_{t+1}}) &= \lambda(K-1) + \lambda_1 \tilde{d}_{t+1,z_{t+1}} - \lambda_2 \log(a_{z_t,z_{t+1}}) \\
&+ \lambda_3\, \tau_{z_{t+1}} + \text{dist}(\boldsymbol{\xi}_{t+1}, \tilde{\boldsymbol{\mu}}_{t+1,z_{t+1}}, \tilde{\boldsymbol{U}}_{t+1,z_{t+1}}^{\tilde{d}_{t+1,z_{t+1}}})^2 \\
&\leq \mathcal{L}(z_{t+1}, \tilde{d}_{t,z_{t+1}}, \tilde{\boldsymbol{\mu}}_{t,z_{t+1}}, \tilde{\boldsymbol{U}}_{t,z_{t+1}}^{\tilde{d}_{t,z_{t+1}}}, a_{t,z_t,z_{t+1}}),
\end{aligned}$$

where $\tilde{\boldsymbol{U}}_{t+1,z_{t+1}}^{\tilde{d}_{t+1,z_{t+1}}}$ corresponds to the basis vectors of the resulting $\tilde{\boldsymbol{\Sigma}}_{t+1,z_{t+1}}$ and $\tilde{d}_{t+1,z_{t+1}} = \min_j d_{t+1,z_{t+1}}^{(j)}$, i.e., the product of Gaussians subspace dimension is defined by the minimum of corresponding subspace dimensions of the Gaussians in P frames for the z_{t+1} mixture component.

5.7 Experiments, Results and Discussions

In this section, we first evaluate the performance of the SOSC model to encode the synthetic data with a 3-dimensional illustrative example followed by its capability to scale in high dimensional spaces. We then consider a real-world application of learning robot manipulation tasks for semi-autonomous teleoperation with the proposed task-parameterized SOSC algorithm. The goal is to assess the performance of the SOSC model to handle noisy online time-series data in a parsimonious manner.

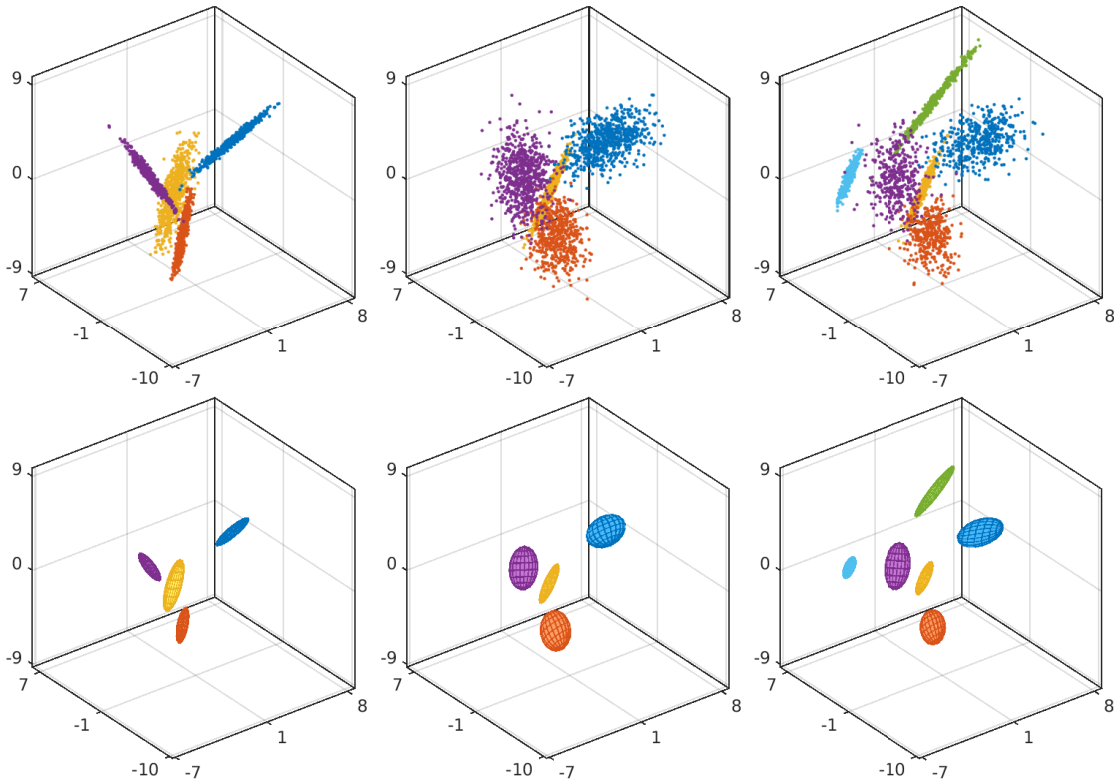


Figure 5.5: Online streaming data generated from a left-right cyclic HSMM on top and encoding with the SOSC model on bottom: *(left)* $K = 4$, d_k is randomly chosen, $t = 1 \ldots 2500$, *(middle)* $K = 4$, $d_k = D - d_k$, $t = 2501 \ldots 5000$, *(right)* $K = 6$, d_k is the same as before, $t = 5001 \ldots 7500$.

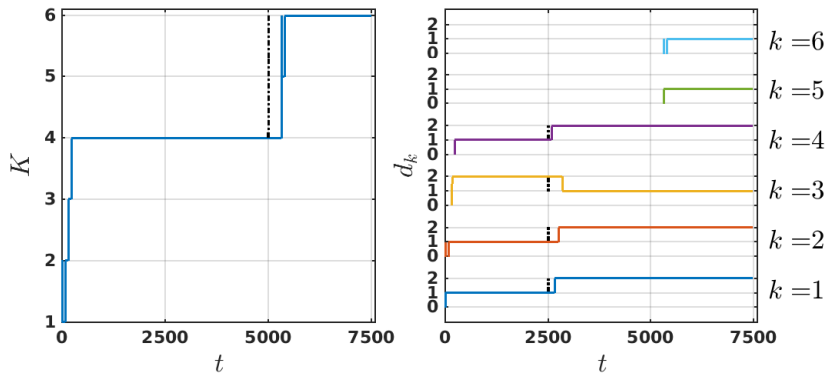


Figure 5.6: Evolution of K and d_k with number of datapoints.

5.7.1 Synthetic Data

5.7.1.1 Non-Stationary Learning with 3-Dimensional Data

We consider a 3-dimensional stream of datapoints $\boldsymbol{\xi}_t \in \mathbb{R}^3$ generated by stochastic sampling from a mixture of different clusters that are connected in a left-right cyclic HSMM. The centers of the clusters are successively drawn from the interval $[-5, 5]$ such that the next cluster is at least $4\sqrt{D}$ units farther than the existing set of clusters. Subspace dimension of each cluster is randomly chosen to lie up to $(D-1)$ dimensions (a line or a plane for 3-dimensions), and the basis vectors are sampled randomly in that subspace. Duration steps in a given state are sampled from a uniform distribution in the interval $[70, 90]$ after which the data is subsequently generated from the next cluster in the model in a cyclic manner. A white noise of $\mathcal{N}(\mathbf{0}, 0.04\boldsymbol{I})$ is added to each sampled data point. Model learning is divided in three stages: 1) for the first 2500 instances, the number of clusters is set to 4 and the subspace dimension of each cluster is fixed, 2) for the subsequent 2500 instances, we change the subspace dimension of each cluster to $(D - d_k)$ for $k = 1 \ldots K$ (for example, a line becomes a plane), while keeping the same number of clusters, and 3) two more clusters are then added in the mixture model for the next 2500 instances without any change in the subspace dimension of the previous clusters. The parameters are defined as $\{\lambda = 3.6, \lambda_1 = 0.35, \lambda_2 = \lambda_3 = 0.025, \sigma^2 = 0.15, b_m = 50\}$. The weights of the parameter update are based on eligibility traces as in Eq. (5.16) with a discount factor of 0.995.

Results of the learned model are shown in Fig. 5.5. We can see that the SOSC model is able to efficiently encode the number of clusters and the subspace dimension of each cluster in each stage of the learning process. The model projects each datapoint in the subspace of the nearest cluster, in contrast to the K-means clustering based on the Euclidean distance metric only. The model is able to adapt the subspace dimension of each cluster in the

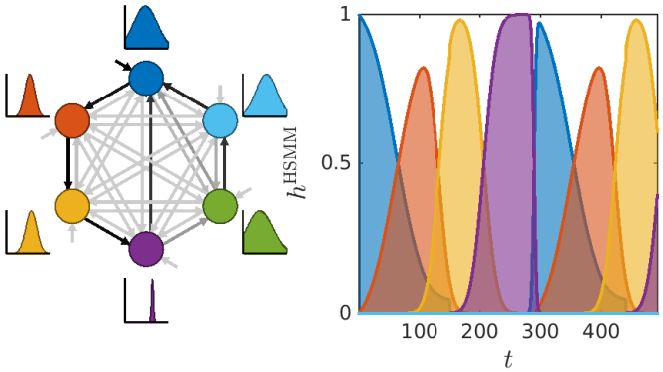


Figure 5.7: *(left)* Learned HSMM transition matrix and state duration model representation with $s^{\max} = 150$, *(right)* rescaled forward variable, $h_{t,i}^{\text{HSMM}} = \frac{\alpha_{t,i}^{\text{HSMM}}}{\sum_{k=1}^{K} \alpha_{t,k}^{\text{HSMM}}}$, sampled from initial position.

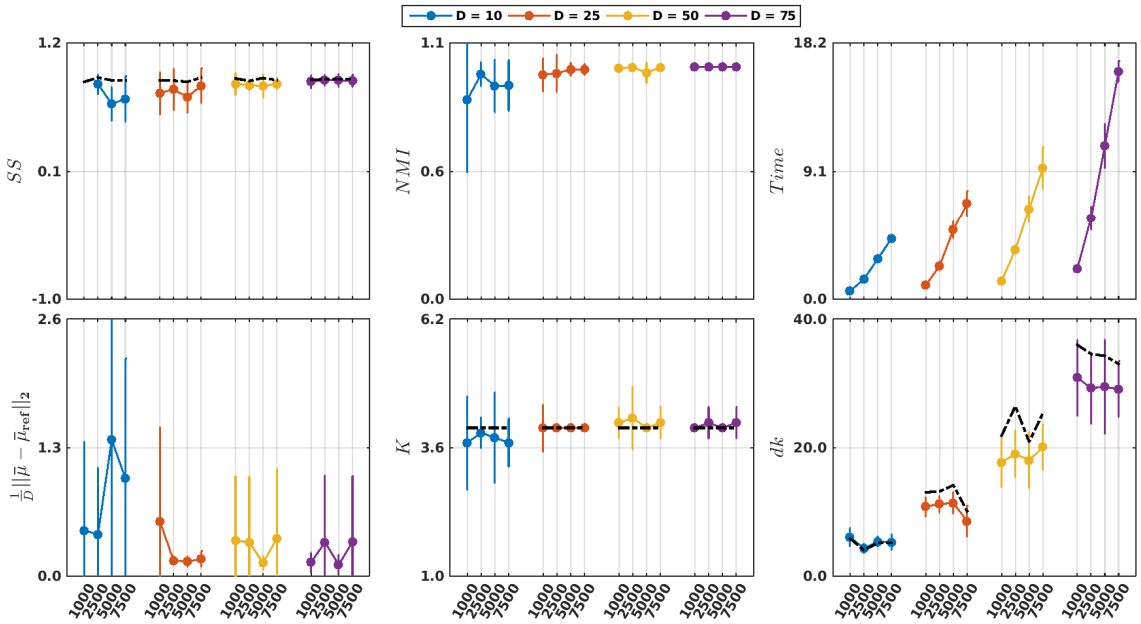


Figure 5.8: SOSC model evaluation to encode synthetic high-dimensional data. Results are averaged over 10 iterations. Black dotted lines indicate the reference value: *(top-left)* silhouette score (SS), *(top-middle)* normalized mutual information score (NMI), *(top-right)* time in seconds, *(bottom-left)* average distance between learned cluster means and ground truth, *(bottom-middle)* number of clusters, *(bottom-right)* average subspace dimension across all clusters.

second stage of the learning process and subsequently incorporate more clusters in the final stage with the non-stationary data. Fig. 5.6 shows the evolution of the number of clusters and the subspace dimension of each cluster with the streaming data. Note that the encoding problem is considerably hard here as the model starts with one cluster only and adapts during the learning process. Clusters that evolve to come closer than a certain threshold are merged during the learning process. Fig. 5.7 shows the graphical model representation of the learned HSMM with the state transitions and the state duration model, along with a sample of the forward variable generated from the initial position.

5.7.1.2 Stationary Learning with High-Dimensional Data

In this experiment, we sample the data from a stationary distribution corresponding to the first stage of the previous example where $K = 4$ and the subspace of each cluster does not change in the streaming data. Dimensionality of the data is successively chosen from the set $D = \{10, 25, 50, 75\}$, and the number of instances are varied for each dimension from the set $T = \{1000, 2500, 5000, 7500\}$. Parameter λ is experimentally selected for each experiment to achieve satisfying results, and the weights of the parameter update are linearly incremented for each cluster. Fig. 5.8 shows the performance of the SOSC model to encode data in high dimensions averaged over 10 iterations. Our results show that the algorithm yields a compact encoding as indicated by high values of the average *silhouette score* (SS),[6] and the *normalized mutual information* (NMI) score,[7] while being robust to the intrinsic subspace dimension of the data and the number of clusters.

5.7.2 Tracking Screwdriver Target and Hooking Carabiner Examples

We are interested in learning the task-parameterized SOSC model online from the teleoperator demonstrations and provide a probabilistic formulation to predict his/her intention while performing the task. The model is used to recognize the intention of the

[6]Silhouette score (SS) measures the tightness of a cluster relative to the other clusters without using any labels,

$$\mathrm{SS}_i \triangleq \frac{b_i - a_i}{\max\{a_i, b_i\}}, \qquad \mathrm{SS}_i \in [-1, 1],$$

where a_i is the mean distance of $\boldsymbol{\xi}_i$ to the other points in its own cluster, and b_i is the mean distance of $\boldsymbol{\xi}_i$ to the points in the closest 'neighbouring' cluster.

[7]Normalized mutual information (NMI) is an extrinsic information-theoretic measure to evaluate the alignment between the assigned cluster labels $\mathcal{Z}$ and the ground truth cluster labels $\mathcal{X}$,

$$\mathrm{NMI}(\mathcal{Z}, \mathcal{X}) \triangleq \frac{I(\mathcal{Z}, \mathcal{X})}{[H(\mathcal{Z}) + H(\mathcal{X}))]/2}, \quad \mathrm{NMI}(\mathcal{Z}, \mathcal{X}) \in [0, 1],$$

where $I(\mathcal{Z}, \mathcal{X})$ is the mutual information and $H(\mathcal{X})$ is the entropy of cluster labels $\mathcal{X}$.

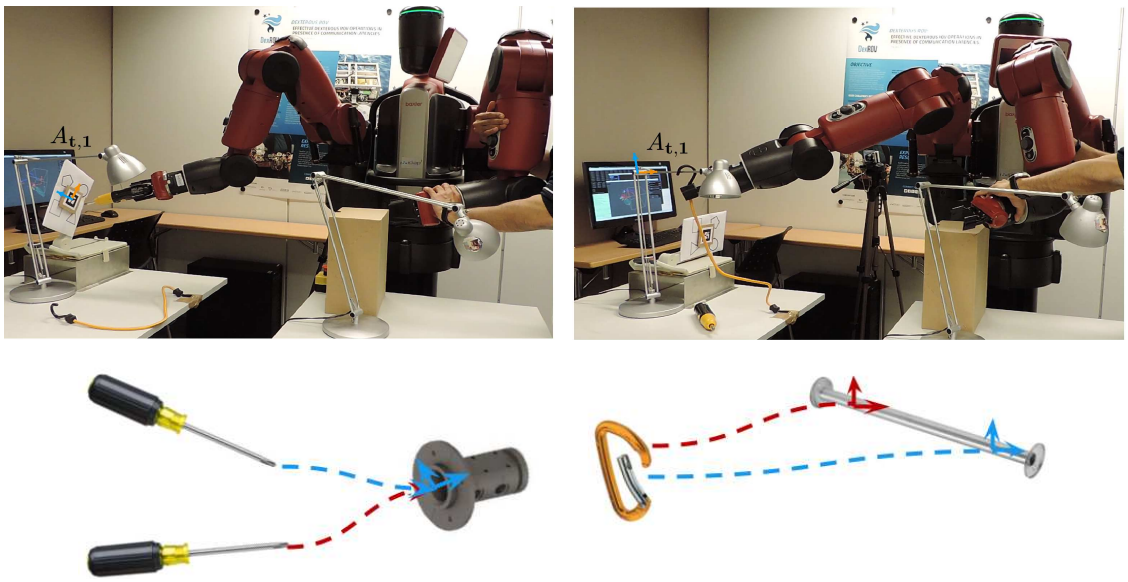


Figure 5.9: Semi-autonomous teleoperation with the Baxter robot for guided assistance of manipulation tools: *(left)* screwdriving with a frame attached to the movable target, *(right)* hooking a carabiner with a frame attached to a rotatable rod. The target for screwdriver is a given pose, while the carabiner can be hooked anywhere along the rod from different initial conditions.

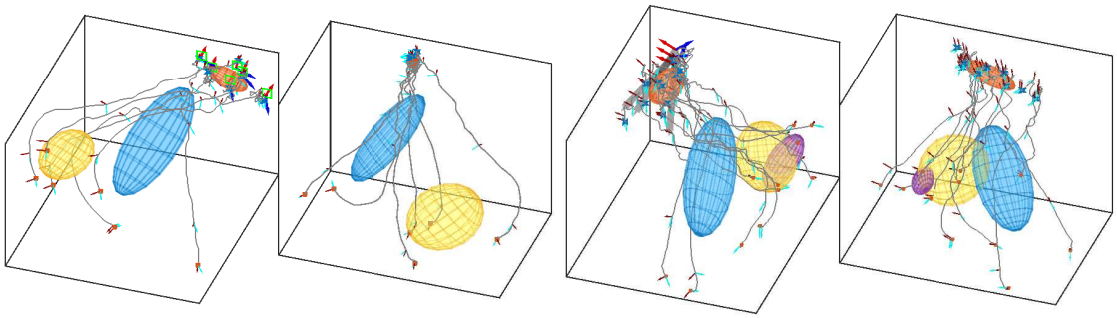


Figure 5.10: Joint distribution of the task-parameterized SOSC model for guided assistance in the screwdriving task *(left)* and hooking a carabiner task *(right)*. For each task, demonstrations and model with respect to the input part of the coordinate system on *(left)*, and with respect to the output part of the coordinate system on *(right)*.

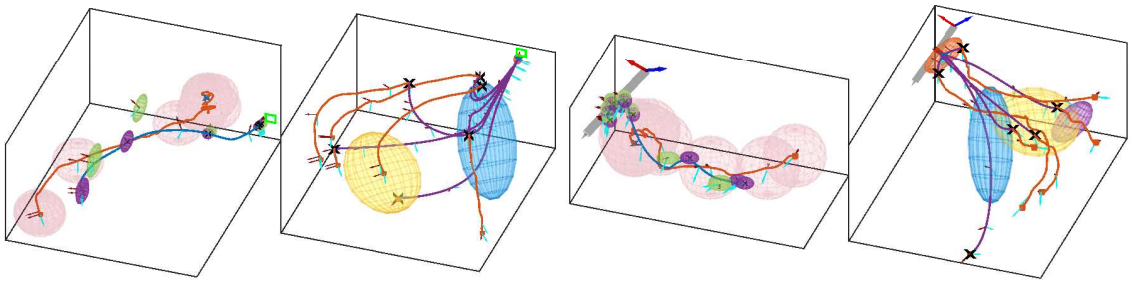


Figure 5.11: Semi-autonomous teleoperation for a new target pose with a screwdriver *(left)* and a carabiner *(right)*. For each task, the shared control example is on *(left)* and the autonomous control example is on *(right)*. In the shared control example, the teleoperator demonstration (in red) strays away from the target pose, while the corrected trajectory (in blue) reaches the target pose. Desired state is shown in purple, teleoperator state in red, and predicted state in green. In the autonomous control example, the arm movement is randomly switched (marked with a cross) from direct control (in red) to autonomous control (in purple) in which the learned model is used to generate the movement to the target pose.

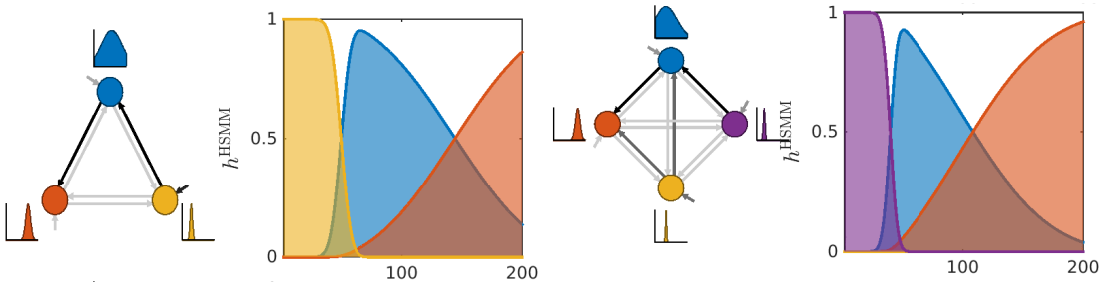


Figure 5.12: HSMM graphical model representation ($s^{\max} = 150$) on *(left)* along with evolution of the rescaled forward variable on (*right*). The left two figures are for screwdriving task and the right two figures are for hooking a carabiner task.

teleoperator, and synthesize motion on the remote end to perform manipulation tasks in a semi-autonomous manner. Two didactic examples of manipulation tasks are incrementally learned for guided assistance: target tracking with a screwdriver and hooking a carabiner. The tasks are selected to reflect typical constraints encountered in daily life. The screwdriver task requires the robot to be invariant to the target pose, while the carabiner can be hooked anywhere along the rod (see also our work on [Havoutis 2016] for application to hot-stabbing task akin to peg-in-a-hole task).

We represent the state of the environment as $\boldsymbol{\xi}_t = \left[\boldsymbol{\xi}_t^{\mathcal{I}^\top} \ \boldsymbol{\xi}_t^{\mathcal{O}^\top}\right]^\top$ with $\boldsymbol{\xi}_t^{\mathcal{I}} \in \mathbb{R}^7$ and $\boldsymbol{\xi}_t^{\mathcal{O}} \in \mathbb{R}^7$ representing respectively the state of the teleoperator arm and the state of the teleoperator arm observed in the coordinate system of the target pose of the tool (screwdriver/carabiner). The state of the teleoperator arm is represented by the position $\boldsymbol{x}_t^p \in \mathbb{R}^3$ and the orientation $\boldsymbol{\varepsilon}_t^o \in \mathbb{R}^4$ of the teleoperator arm end-effector in their respective coordinate systems with $D = 14$. We attach a frame $\{\boldsymbol{A}_{t,1}, \boldsymbol{b}_{t,1}\}$ to the target pose of the tool. Note that the frame has two components, the input component represents the teleoperator pose in the global frame corresponding to $\boldsymbol{\xi}_t^{\mathcal{I}}$, while the output component maps the teleoperator state with respect to the target pose corresponding to $\boldsymbol{\xi}_t^{\mathcal{O}}$.

Based on the learned joint distribution of the task-parameterized SOSC model, we seek to recognize the intention of the teleoperator and subsequently correct the current state of the teleoperated arm by estimating the conditional distribution $\mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{O}}|\boldsymbol{\xi}_t^{\mathcal{I}})$. The intention here refers to the cluster or the mixture component to which the teleoperator belongs. The correction is time-independent and control is shared between the teleoperator and the remote arm. In case of communication disruptions, we solicit the model to generate the movement on the remote arm in a time-dependent autonomous manner. After the task is completed, the arm comes back to the desired position as estimated under shared control. We provide the details of shared and autonomous control formulations of the generative model in the next chapter.

We collect 6 kinesthetic demonstrations for screwdriving with the initial pose of the target rotated/translated in the successive demonstrations, and perform 11 demonstrations of hooking a carabiner at various places on the rod for 3 different rotated configurations of the rod segment. Demonstrations are subsampled around 7 Hz and limited to 200 datapoints for each demonstration. The parameters are defined as $\{\lambda = 0.65, \lambda_1 = 0.03, \lambda_2 = 0.001, \lambda_3 = 0.04, \sigma^2 = 2.5 \times 10^{-4}, \kappa^2 = 0.01\}$.

Results of the task-parameterized SOSC model for the two tasks are shown in Fig. 5.10. We observe that the model exploits the variability in the demonstrations to statistically encode different phases of the task in the joint distribution. Demonstrations corresponding to

Table 5.1: Performance comparison of the SOSC model against parametric batch HSMM models using number of parameters N_p, and the endpoint error between the teleoperated arm and the target. Teleoperation modes are direct control (DC), shared control (SC) and autonomous control (AC). Errors are reported in meters.

Model	N_p	**DC Error**	**SC Error**	**AC Error**
screw-driving task ($K = 3, D = 14$)				
FC-HSMM	372	0.30± 0.17	0.095± 0.025	0.038± 2.5×10^{-5}
ST-HSMM	295		0.094± 0.026	0.037± 1.8×10^{-5}
MFA-HSMM ($d_k = 4$)	267		0.099± 0.022	**0.037**± 7.7×10^{-6}
SOSC ($\bar{d}_k = 3.67$)	**211**		**0.084**± 0.018	0.043± 1.3×10^{-4}
hooking carabiner task ($K = 4, D = 14$)				
FC-HSMM	500	0.10± 0.062	0.081± 0.056	0.099± 0.068
ST-HSMM	332		0.082± 0.058	**0.022**± 2.6×10^{-4}
MFA-HSMM ($d_k = 4$)	360		**0.08**± 0.056	0.037± 8.8×10^{-4}
SOSC ($\bar{d}_k = 4.25$)	**318**		**0.08**± 0.056	0.073± 3.7×10^{-4}

the input component of the frame encode the reaching movement to different target poses with the screwdriver and the carabiner in the global frame, while the output component of the frame represents this movement observed from the viewpoint of the target (respectively shown as converging to a point for the screwdriver and to a line for the carabiner). The learned model for the screwdriving task contains 3 clusters with subspace dimensions $\{4, 3, 4\}$, while the carabiner task model contains 4 clusters with subspace dimensions $\{5, 5, 4, 3\}$.

Fig. 5.11 (*left*) shows how the model adjusts the movement of the teleoperator based on his/her current state in a time-independent manner. When the teleoperator is away from the target, the variance in the output conditional distribution is high and the desired state is closer to the teleoperator as in direct teleoperation. As the teleoperator moves closer to the target and visits low variance segments, the desired state moves closer to the target as compared to the teleoperator. Consequently, the shared control formulation corrects the movement of the teleoperator when the teleoperator is straying from the target. Table 5.1 shows the performance improvement of shared control over direct control where the endpoint error is reduced from 0.3 to 0.084 meters for the screwdriving task, and from 0.1 to 0.08 meters for the carabiner task. Error is measured at the end of the demonstration from the end-effector of the teleoperated arm to the target of the screwdriver, and to the rod segment for hooking the carabiner.

To evaluate the autonomous control mode of the task-parameterized SOSC model, the teleoperator performs 6 demonstrations and switches to the autonomous mode randomly while performing the task. The teleoperated arm evaluates the current state of the task and generates the desired sequence of states to be visited for the next T steps using the forward variable of HSMM (see Fig. 5.12). Fig. 5.11 (*right*) shows that the movement of the robot converges to the target from different initial configurations of the teleoperator. The obtained results are repeatable and more precise than the direct and the shared control results, as shown in Table 5.1. Moreover, the table also compares the performance of the SOSC algorithm against several parametric batch versions of HSMMs with different covariance models in the output state distribution, including full covariance (FC-HSMM), semi-tied covariance (ST-HSMM), and MFA decomposition of covariance (MFA-HSMM). Results of the SOSC model are used as a reference for model selection of the batch algorithms. We can see that the proposed non-parametric online learning model gives comparable performance to other parametric batch algorithms with a more parsimonious representation (reduced number of model parameters).

5.8 Conclusions

Non-parametric online learning is a promising way to adapt the model on the fly with new training data. In this chapter, we have presented online learning algorithms for Bayesian non-parametric mixture models under small variance asymptotics. The resulting scalable online sequence clustering algorithm, obtained by online inference in HDP-HSMM with DP-MPPCA as output state distribution, incrementally groups the streaming data with non-parametric locally linear principal component analysis and encodes the spatio-temporal patterns using an infinite hidden semi-Markov model. Non-parametric treatment gives the flexibility to continuously adapt the model with new incoming data. Learning the model online from a few human demonstrations is a pragmatic approach to teach new skills to robots. The proposed skill encoding scheme is potentially applicable to a wide range of tasks, while being robust to varying environmental conditions with the task-parameterized formulation. We showed the efficacy of the approach to learn manipulation tasks online for semi-autonomous teleoperation, and assist the operator with shared control and/or autonomous control in performing remote manipulation tasks.

Chapter 6

Manipulation Assistance in Teleoperation

Contents

6.1 Teleoperation Scenario - An Illustrative Example 130
6.2 High Level Architecture 132
6.2.1 Cognitive Engine 132
6.2.2 Proxy Cognitive Engine 134
6.2.3 Communication Interfaces 135
6.3 Intention Recognition and Manipulation Assistance 135
6.3.1 Time-Independent Shared Control 138
6.3.2 Time-Dependent Autonomous Control 139
6.4 Comparison with Virtual Fixtures 140
6.5 Experiments, Results and Discussions 141
6.5.1 Task Performance Error 143
6.5.2 Robustness to Different Environments 144
6.5.3 Execution Time 145
6.6 Conclusion 145

This chapter exploits the use of task-parameterized generative models for providing assistance to the teleoperator in performing remote manipulation tasks. We present *time-independent shared control* and *time-dependent autonomous control* formulations of the hidden semi-Markov model that captures the intention of the teleoperator and subsequently, provides manipulation assistance to the teleoperator [Tanwani 2017]. In the shared control

mode, the model corrects the remote arm movement based on the current state of the teleoperator; whereas in the autonomous control mode, the model generates the movement of the remote arm for autonomous task execution. We show the formulation of the model with well-known *virtual fixtures* [Abbott 2007] and provide comparisons to benchmark our approach. Teleoperation experiments with the Baxter robot reveal that the proposed methodology improves the performance of the teleoperator and caters for environmental differences and communication delays in performing remote manipulation tasks.

We are interested in performing dexterous manipulation tasks in remote challenging underwater environments within the DexROV project [Gancet 2015]. Large communication delays with satellite communication render direct teleoperation infeasible, thereby, requiring semi-autonomous capabilities of the remotely operated arm to carry out manipulation tasks. The operational costs are significantly reduced by moving the teleoperation personnel from the vessel to operate the vehicle from a remote facility. We use the two-armed Baxter robot as a mock-up of the teleoperation system, i.e., one arm becomes the input device for the teleoperator, and the other one is used for performing the manipulation task. The operator controls/teleoperates the remote arm with a simulated delay using the other arm by getting visual feedback from the remote arm. A set of kinesthetic demonstrations of the teleoperator is used to teach the robot how to perform each task. We seek to leverage upon our previous work on probabilistic generative models [Tanwani 2016a, Tanwani 2016c, Tanwani 2016b] to understand the intention of the teleoperator and assist the movement on the robot side under varying environmental situations.

6.1 Teleoperation Scenario - An Illustrative Example

Consider a simple task of grasping an object on the remote site by teleoperation. The task is demonstrated on the teleoperator site from different initial configurations of the arm and the object. After learning the model from a few demonstrations, the model parameters are passed to the remote site during the start of the mission (implemented as a ROS service). During teleoperation, the teleoperator arm data is continuously streamed to the remote site, while the remote robot arm data and the object description (reference frames described as task parameters) are sent back to the teleoperator side. To simulate communication latency in teleoperation, data is buffered on both the teleoperation and the remote sites. Fixed time delays of up to 2 seconds are introduced, under which the teleoperator perceives the object with delayed feedback.

The teleoperator has two modes of assistance as illustrated in Fig. 6.1: 1) shared control, and 2) autonomous control. Shared control continuously adjusts/corrects the robot

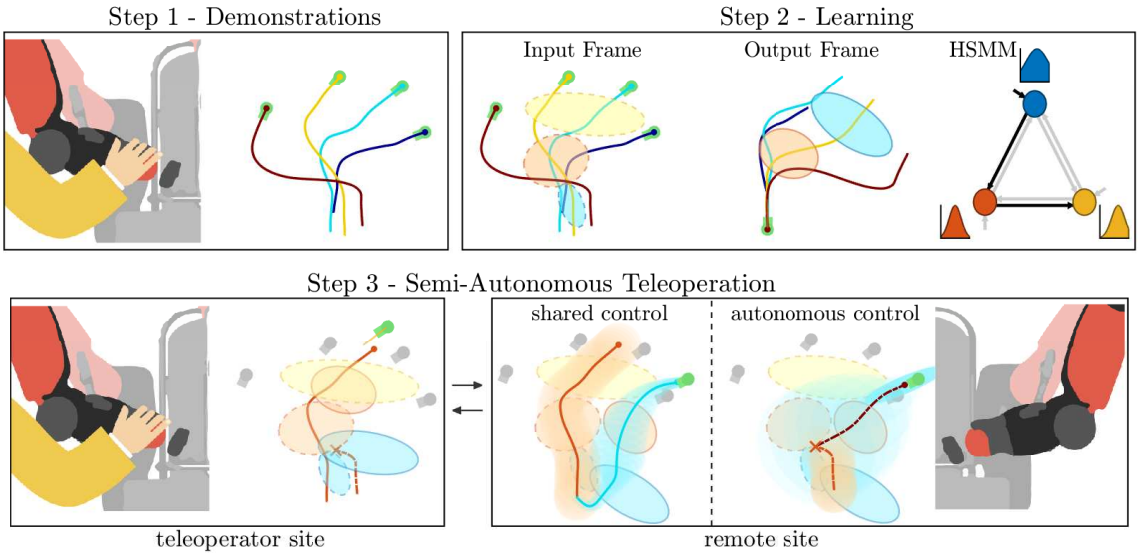


Figure 6.1: Semi-autonomous teleoperation framework: Step 1) the teleoperator provides a few demonstrations of the manipulation task under different object positions shown in green (the green targets depict the end of the demonstrations); Step 2) a task-parameterized HSMM is learned, with the input frame of reference representing the demonstrations in the global coordinate system, and the output frame of reference representing the demonstrations in the coordinate system attached to the object (Gaussian depicted as an ellipse represents the emission distribution of a state; the graphical representation of HSMM shows transition among states and the state duration modeled with a Gaussian); Step 3) *(left)* the teleoperator performs the imprecise movement (in orange) to grasp the perceived object in green, *(right)* the shared control mode corrects the movement of the robot (in blue) locally in accordance with the actual object position on the remote site, while the autonomous control mode generates the movement to the object (in dark red) after the teleoperator switches to the autonomous mode (marked with a cross). Note that the output frame component adapts the model locally in accordance with the object.

movement given the teleoperator arm data based on the learned model that locally adapts according to the object position [Vogel 2016]. The model exploits the variability observed in the teleoperator demonstrations. Where the variance is high such as away from the object, the correction is mild, whereas for low variance regions close to the object, the model strongly corrects the remote arm to track the object. Supervisory control gives the robot more autonomy as the model detects the state of the task and generates the remote arm movement to accomplish the task. Fig. 6.2 shows the setup used to deploy the proposed semi-autonomous teleoperation of ROV in real underwater environments.

Figure 6.2: Teleoperation setup to control ROV: *(left)* exoskeleton and the virtual reality headset worn by the teleoperator, *(middle)* third person view of the teleoperator in the virtual reality environment, *(right)* ROV with manipulator controlled by the teleoperator.

6.2 High Level Architecture

Our high level architecture of controlling distant robots by semi-autonomous teleoperation consists of two modules: ***cognitive engine*** and ***proxy cognitive engine***. The cognitive engine provides a method for teleoperation to cope with long transmission delays while assisting the teleoperator in performing remote manipulation tasks. The engine is comprised of a library of task models. The model parameters of each task are learned from the demonstrations provided by the teleoperator. The set of task models are concatenated together and used – during the mission and in subsequent missions – to provide assistance to the teleoperator in performing remote manipulation tasks. The aim is to reduce the cognitive load of the teleoperator for repetitive or well structured tasks, while also increasing efficiency and accuracy by closing a local sensory feedback loop. The cognitive engine is split between the teleoperator site and the remote site, described using the following subsystems:

6.2.1 Cognitive Engine

The cognitive engine resides on the teleoperator site. It is mainly responsible for handling the input provided by the teleoperator. Within the DexROV project, the teleoperator's input comes from external interfaces, namely the exoskeleton and the virtual reality environment used to immerse the teleoperator with the remote environment.

The cognitive engine has two phases of operation: a model learning phase, and a semi-autonomous teleoperation phase.

6.2.1.1 Model Learning

The model learning phase is offline. In this phase, the teleoperator provides demonstrations of the task to be performed. The environmental setup of each task is either physically constructed or simulated in the virtual reality in which the teleoperator can interact with the object(s) in the environment. The teleoperator performs a few demonstrations of the underlying task, and the cognitive engine learns a task-parameterized generative model used online during the mission. Task by task a library is built that can serve multiple future missions. Before the start of a new mission, the teleoperator loads the model of the task from the library which has already been performed. The protocol of the teleoperator is defined as follows.

The teleoperator prepares for a new mission by defining all the important coordinate systems/frames of reference in the environment. The frames describe the coordinate systems with respect to which the movement needs to be adapted. For each task, the teleoperator provides the task description by defining the *task name*, *input* and *output* components of the frame relevant for the task. After initializing the task, the teleoperator typically performs $4 - 10$ demonstrations each containing $50 - 200$ datapoints. The operator learns a task-parameterized HSMM from the demonstrations and verifies the shared and autonomous control modes of teleoperation. If the teleoperator is satisfied with the assistance behaviour of the model, he/she can save the model in the database. Alternatively, he/she can either add more demonstrations or change the hyperparameters of the algorithm. If the model and/or the demonstrations are not satisfactory, the teleoperator may delete the model. The saved models are associated with a unique task id which can be loaded again for future missions.

6.2.1.2 Manipulation Assistance

The learned model is used to assist the teleoperator online while performing remote manipulation tasks. The cognitive engine receives the teleoperator input from external sources such as the exoskeleton and the virtual reality environment at each time step. The model recognizes the intention of the teleoperator and provides assistance in performing remote manipulation tasks in a time-independent shared control or time-dependent autonomous control manner.

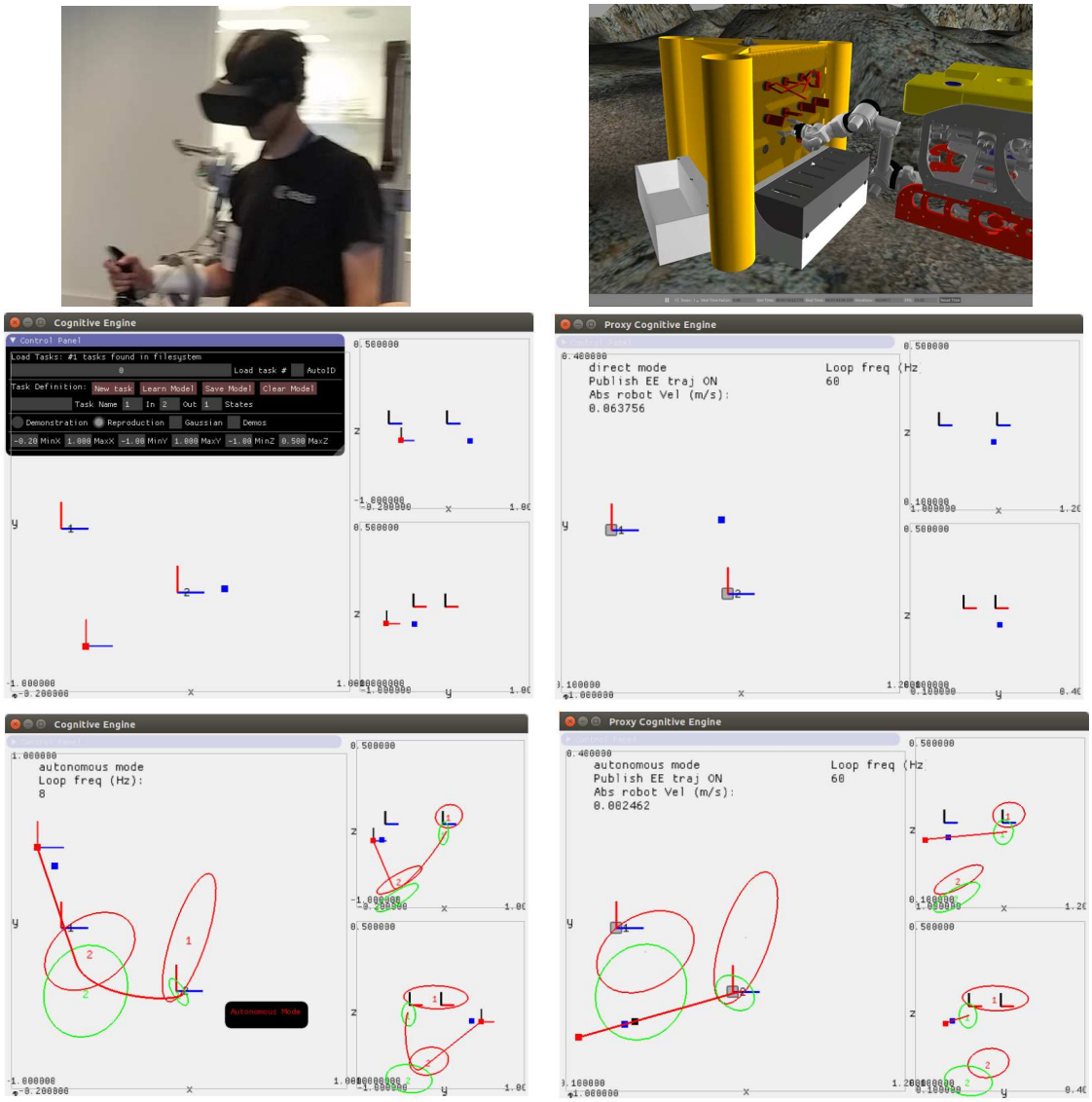


Figure 6.3: Cognitive engine on *(left)* is used for handling input demonstrations from the teleoperator and proxy cognitive engine on *(right)* locally adjusts the teleoperator input to provide assistance in performing remote manipulation tasks. The top figure shows the teleoperator and the remote control sites, the middle figure shows the control panel, whereas the bottom figure displays the congitive engine simulator instance in autonomous mode.

6.2.2 Proxy Cognitive Engine

As the name indicates, the proxy cognitive engine is a copy of the cognitive engine that resides on the remote site. Prior to the start of the mission, the model parameters are transmitted from the cognitive engine to the proxy cognitive engine on the remote site. The

perception system on the remote site locally updates the coordinate systems to describe the environmental situation and transmits this information back to the cognitive engine over satellite communication. The proxy cognitive engine receives delayed input from the teleoperator site about the current state of the teleoperator which is locally adjusted in accordance with the task model to provide manipulation assistance in performing remote manipulation tasks. Fig. 6.3 shows a minimalist cognitive engine simulator interface split into the cognitive engine side on left and the proxy cognitive engine on right. During autonomous mode, the simulator displays the trajectories of the teleoperator and the robot arm on the cognitive engine and the proxy side, respectively.

6.2.3 Communication Interfaces

The communication between the cognitive engine, proxy cognitive engine and external interfaces is done using ROS messages and services. Fig. 6.4 summarizes the communication interfaces for the cognitive engine. An example of the communication interfaces for the valve turning task is shown in Fig. 6.5.

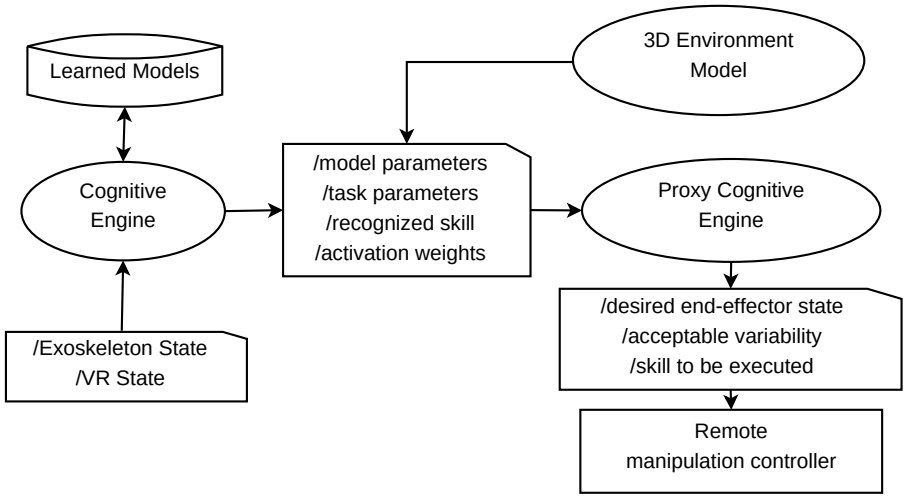


Figure 6.4: ROS based system representation of the cognitive engine.

6.3 Intention Recognition and Manipulation Assistance

We denote an observation of the teleoperator arm as $\boldsymbol{\xi}_t \in \mathbb{R}^D$ with $\boldsymbol{\xi}_t = \left[\boldsymbol{\xi}_t^{\mathcal{I}^\top} \; \boldsymbol{\xi}_t^{\mathcal{O}^\top}\right]^\top$ where $\boldsymbol{\xi}_t^{\mathcal{I}}$ and $\boldsymbol{\xi}_t^{\mathcal{O}}$ respectively represent the pose of the end-effector of the teleoperator arm at time t in a global coordinate system and the same pose observed with respect

Cognitive Engine Publishers:
(1) Proxy Cognitive Engine
- model_parameters: $\{\mu_1^{(1)}, \Sigma_1^{(1)}\}, \{\mu_2^{(1)}, \Sigma_2^{(1)}\}$
  (Gaussians in each frame, transferred as ROS Service at start of mission)
- exoskeleton end-effector state: geometry_msgs/Pose and geometry_msgs/Twist

Cognitive Engine Subscribers:
(1) 3D Environment
- relevant objects: **[Valve centroid position and orientation o_1]**
- landmarks/points of interest: **[Valve corner points for length/shape of valve]**

(2) Teleoperator Exoskeleton
- exoskeleton: end-effector state: geometry_msgs/Pose and geometry_msgs/Twist

(5) Semi-Autonomous ROV Manipulation Robot Arm
- Remote robot_arm end-effector state: geometry_msgs/Pose and geometry_msgs/Twist

Proxy Cognitive Engine Subscribers:
(1) Cognitive Engine
- model_parameters: $\{\mu_1^{(1)}, \Sigma_1^{(1)}\}, \{\mu_2^{(1)}, \Sigma_2^{(1)}\}$
  (Gaussians in each frame, received as ROS Service at start of mission)
- exoskeleton end-effector state: geometry_msgs/Pose and geometry_msgs/Twist

Proxy Cognitive Engine Publishers:
(1) Semi-Autonomous ROV Manipulation Robot Arm
- robot_arm, desired end-effector state $\hat{x}_t$
  (desired position and velocity with 3D position and quaternion)
- robot_arm, task variability / coordination / precision: $\hat{\Sigma}_{x_t}$
  (full covariance matrix including coordination information)
- task currently executed
  (for example, [0 1 0 0 0 0] with 1 corresponding to valve opening task)

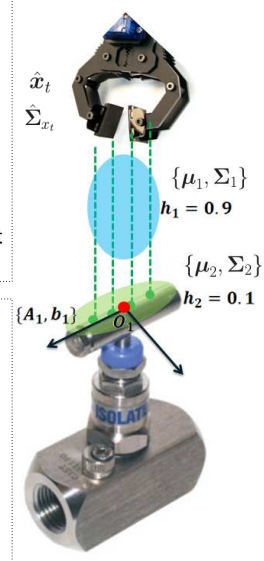


Figure 6.5: ROS publishers and subscribers for valve opening task encoded with 1 frame and 2 mixture components. h_1 and h_2 respectively denote the activation weights of the mixture components based on the current position of the robot arm.

to another coordinate describing the current context or situation (superscripts $\imath$ and o represent the input and the output components). The aim of augmenting the teleoperator pose with different coordinate systems is to couple the movement of the teleoperator arm with external environmental variables, i.e., we learn the mapping between the teleoperator pose in two reference frames: in a global frame and in the object frame, modeled as a joint distribution. We assume that the reference frames are specified by the user, based on prior knowledge about the carried out task. Typically, reference frames will be attached to objects, tools or locations that could be relevant in the execution of a task.

The pose of the teleoperator arm describes the position $\boldsymbol{x}_t^p \in \mathbb{R}^3$ and the unit quaternion orientation $\boldsymbol{\varepsilon}_t^o \in \boldsymbol{S}^3$ of the end-effector ($D = 14$ with 7 dimensional pose of the input dimension and 7 dimensional pose of the output dimension observed with respect to task

parameters). We represent the task parameters with P coordinate systems, defined by the reference frames $\{\boldsymbol{A}_j, \boldsymbol{b}_j\}_{j=1}^P$ where $\boldsymbol{A}_j$ denotes the orientation of the frame and $\boldsymbol{b}_j$ represents the origin of the frame. The frame $\{\boldsymbol{A}_j, \boldsymbol{b}_j\}$ is described by

$$\boldsymbol{A}_j = \begin{bmatrix} \boldsymbol{I}^{\mathcal{I}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_j^{\mathcal{O}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \mathcal{E}_j^{\mathcal{O}} \end{bmatrix}, \boldsymbol{b}_j = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{p}_j^{\mathcal{O}} \\ \boldsymbol{0} \end{bmatrix}, \tag{6.1}$$

where $\boldsymbol{p}_j^{\mathcal{O}} \in \mathbb{R}^3$, $\boldsymbol{R}_j^{\mathcal{O}} \in \mathbb{R}^{3\times3}$, $\mathcal{E}_j^{\mathcal{O}} \in \mathbb{R}^{4\times4}$ denote the Cartesian position, the rotation matrix and the quaternion matrix of the j-th frame, respectively.

The observation sequence $\{\boldsymbol{\xi}_t\}_{t=1}^T$ of T datapoints, observed from the perspective of different coordinate systems, forms a third order tensor dataset $\{\boldsymbol{\xi}_t^{(j)}\}_{t,j=1}^{T,P}$ with $\boldsymbol{\xi}_t^{(j)} = \boldsymbol{A}_j^{-1}(\boldsymbol{\xi}_t - \boldsymbol{b}_j)$. This dataset is used to train a task-parameterized HSMM with K hidden states represented by the parameter set $\theta_h = \left\{\Pi_i, \{a_{i,m}\}_{m=1}^K, \{\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\}_{j=1}^P, \mu_i^S, \Sigma_i^S\right\}_{i=1}^K$. The parameter set can be learned in a batch manner as in Chap. 3 or in a non-parametric online manner as in Chap. 5. Additionally, the data can be modeled in latent space to avoid overfitting and exploit coordination patterns based on statistical decomposition of the covariance matrix as seen in Chap. 4.

In the reproduction phase for a given environmental situation represented by the frames $\{\tilde{\boldsymbol{A}}_j, \tilde{\boldsymbol{b}}_j\}_{j=1}^P$, the resulting model parameters $\{\tilde{\boldsymbol{\mu}}_i, \tilde{\boldsymbol{\Sigma}}_i\}$ are obtained by first linearly transforming the Gaussians in the P frames with

$$\mathcal{N}(\tilde{\boldsymbol{\mu}}_i^{(j)}, \tilde{\boldsymbol{\Sigma}}_i^{(j)}) = \mathcal{N}\left(\tilde{\boldsymbol{A}}_j \boldsymbol{\mu}_i^{(j)} + \tilde{\boldsymbol{b}}_j, \tilde{\boldsymbol{A}}_j \boldsymbol{\Sigma}_i^{(j)} \tilde{\boldsymbol{A}}_j^\top\right), \tag{6.2}$$

and then computing the products of the linearly transformed Gaussians for each component with

$$\mathcal{N}(\tilde{\boldsymbol{\mu}}_i, \tilde{\boldsymbol{\Sigma}}_i) \propto \prod_{j=1}^P \mathcal{N}(\tilde{\boldsymbol{\mu}}_i^{(j)}, \tilde{\boldsymbol{\Sigma}}_i^{(j)}), \tag{6.3}$$

$$\tilde{\boldsymbol{\Sigma}}_i = \left(\sum_{j=1}^P \tilde{\boldsymbol{\Sigma}}_i^{(j)}\right)^{-1}, \quad \tilde{\boldsymbol{\mu}}_i = \tilde{\boldsymbol{\Sigma}}_i \sum_{j=1}^P \left(\tilde{\boldsymbol{\Sigma}}_i^{(j)}\right)^{-1} \left(\tilde{\boldsymbol{\mu}}_i^{(j)}\right).$$

We now present the details of two formulations of the learned model to assist the teleoperator in performing remote manipulation tasks: 1) *time-independent shared control*, 2) *time-dependent autonomous control*.

6.3.1 Time-Independent Shared Control

In shared control, we seek to continuously correct the movement of the robot arm according to the learned model given the input data from the teleoperator. We approximate the conditional probability distribution of the teleoperator pose in each output frame component given the current teleoperator pose as $\mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{O}_j}|\boldsymbol{\xi}_t^{\mathcal{I}}) \approx \mathcal{N}(\tilde{\boldsymbol{\mu}}_t^{\mathcal{O}_j}, \tilde{\boldsymbol{\Sigma}}_t^{\mathcal{O}_j})$, based on the joint distribution of the linearly transformed Gaussians $\mathcal{N}(\tilde{\boldsymbol{\mu}}_i^{(j)}, \tilde{\boldsymbol{\Sigma}}_i^{(j)})$. Denoting the block decomposition of the joint distribution as,

$$\tilde{\boldsymbol{\mu}}_i^{(j)} = \begin{bmatrix} \tilde{\boldsymbol{\mu}}_i^{\mathcal{I}_j} \\ \tilde{\boldsymbol{\mu}}_i^{\mathcal{O}_j} \end{bmatrix}, \quad \tilde{\boldsymbol{\Sigma}}_i^{(j)} = \begin{bmatrix} \tilde{\boldsymbol{\Sigma}}_i^{\mathcal{I}_j} & \tilde{\boldsymbol{\Sigma}}_i^{\mathcal{I}\mathcal{O}_j} \\ \tilde{\boldsymbol{\Sigma}}_i^{\mathcal{O}\mathcal{I}_j} & \tilde{\boldsymbol{\Sigma}}_i^{\mathcal{O}_j} \end{bmatrix}, \tag{6.4}$$

the conditional output distribution $\mathcal{N}(\tilde{\boldsymbol{\mu}}_t^{\mathcal{O}_j}, \tilde{\boldsymbol{\Sigma}}_t^{\mathcal{O}_j})$ is approximated using Gaussian mixture regression [Ghahramani 1994],

$$\tilde{\boldsymbol{\mu}}_t^{\mathcal{O}_j} = \sum_{i=1}^{K} h_i(\boldsymbol{\xi}_t^{\mathcal{I}}) \ \hat{\boldsymbol{\mu}}_i^{\mathcal{O}_j}(\boldsymbol{\xi}_t^{\mathcal{I}}), \tag{6.5}$$

$$\tilde{\boldsymbol{\Sigma}}_t^{\mathcal{O}_j} = \sum_{i=1}^{K} h_i(\boldsymbol{\xi}_t^{\mathcal{I}}) \Big(\hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}_j} + \hat{\boldsymbol{\mu}}_i^{\mathcal{O}_j}(\boldsymbol{\xi}_t^{\mathcal{I}}) \hat{\boldsymbol{\mu}}_i^{\mathcal{O}_j}(\boldsymbol{\xi}_t^{\mathcal{I}})^\top \Big) - \tilde{\boldsymbol{\mu}}_t^{\mathcal{O}_j} \tilde{\boldsymbol{\mu}}_t^{\mathcal{O}_j\top}, \tag{6.6}$$

$$\text{with} \quad h_i(\boldsymbol{\xi}_t^{\mathcal{I}}) = \frac{\pi_i \, \mathcal{N}(\boldsymbol{\xi}_t^{\mathcal{I}} | \, \tilde{\boldsymbol{\mu}}_i^{\mathcal{I}_j}, \tilde{\boldsymbol{\Sigma}}_i^{\mathcal{I}_j})}{\sum_k^K \pi_k \, \mathcal{N}(\boldsymbol{\xi}_t^{\mathcal{I}} | \, \tilde{\boldsymbol{\mu}}_k^{\mathcal{I}_j}, \tilde{\boldsymbol{\Sigma}}_k^{\mathcal{I}_j})}, \tag{6.7}$$

$$\hat{\boldsymbol{\mu}}_i^{\mathcal{O}_j}(\boldsymbol{\xi}_t^{\mathcal{I}}) = \tilde{\boldsymbol{\mu}}_i^{\mathcal{O}_j} + \tilde{\boldsymbol{\Sigma}}_i^{\mathcal{O}\mathcal{I}_j} \tilde{\boldsymbol{\Sigma}}_i^{\mathcal{I}_j\,-1} (\boldsymbol{\xi}_t^{\mathcal{I}} - \tilde{\boldsymbol{\mu}}_i^{\mathcal{I}_j}), \tag{6.8}$$

$$\hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}_j} = \tilde{\boldsymbol{\Sigma}}_i^{\mathcal{O}_j} - \tilde{\boldsymbol{\Sigma}}_i^{\mathcal{O}\mathcal{I}_j} \tilde{\boldsymbol{\Sigma}}_i^{\mathcal{I}_j\,-1} \tilde{\boldsymbol{\Sigma}}_i^{\mathcal{I}\mathcal{O}_j}. \tag{6.9}$$

The conditional probability distribution $\mathcal{N}(\tilde{\boldsymbol{\mu}}_t^{\mathcal{O}_j}, \tilde{\boldsymbol{\Sigma}}_t^{\mathcal{O}_j})$ predicts the teleoperator pose according to the learned model and the uncertainty associated with the pose in the given frame $\{\tilde{\boldsymbol{A}}_j, \tilde{\boldsymbol{b}}_j\}$. The conditional probability distributions in all the frames are combined using the product of Gaussians to yield the desired pose at each time instant, $\mathcal{N}(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t) \ \propto \ \prod_{j=1}^{P} \mathcal{N}(\tilde{\boldsymbol{\mu}}_t^{\mathcal{O}_j}, \tilde{\boldsymbol{\Sigma}}_t^{\mathcal{O}_j})$ (see Eq. (6.3)). Note that the variance of the resulting product of Gaussians determines the trade-off between direct teleoperation and correction applied by the model. If the variance is low, the correction is strong and the robot arm follows the model better than the teleoperator. A similar variance based shared control architecture has also been adopted by authors in [Abi-Farraj 2017].

6.3.2 Time-Dependent Autonomous Control

Continuously operating the remote arm for routine tasks can be cumbersome for the teleoperator, especially in the presence of communication latency. In such a situation, the teleoperator may switch at any point in time t_o to the autonomous control mode upon which the robot arm recursively re-plans and executes the task for the next T steps. When the task is accomplished or the communication channel is re-established, the operator switches back to the direct/shared control upon which the robot arm returns to the desired teleoperated state.

The input part of the learned model is used to recognize the most likely state of the task at t_o given the teleoperator pose $\boldsymbol{\xi}_t^{\mathcal{I}}$. The desired movement sequence is then computed with the help of the forward variable of HSMM as seen in Sec. 3.2.3. The forward variable is initialized with the current state of the task $\boldsymbol{\xi}_{t_o}$ using $\alpha_{t_o,i}^{\text{HSMM}} = \frac{\pi_i \mathcal{N}(\boldsymbol{\xi}_{t_o}|\tilde{\boldsymbol{\mu}}_i,\tilde{\boldsymbol{\Sigma}}_i)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{\xi}_{t_o}|\tilde{\boldsymbol{\mu}}_k,\tilde{\boldsymbol{\Sigma}}_k)}$, and is subsequently used to plan the movement sequence for the next T steps with $t=(t_o+1)\dots T$. This is used to retrieve a stepwise reference trajectory $\mathcal{N}(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t)$ from the state sequence z_t computed from the forward variable, with

$$z_t = \arg\max_i \ \alpha_{t,i}^{\text{HSMM}}, \quad \hat{\boldsymbol{\mu}}_t = \tilde{\boldsymbol{\mu}}_{z_t}^{\mathcal{O}}, \quad \hat{\boldsymbol{\Sigma}}_t = \tilde{\boldsymbol{\Sigma}}_{z_t}^{\mathcal{O}}. \tag{6.10}$$

The desired pose in the shared control mode or the stepwise desired sequence of poses in the autonomous control mode is respectively followed with an infinite or a finite horizon discrete-time linear quadratic regulator, see Sec. 3.3.2 for details. The cost function minimized during tracking with our defined state variables is expressed as

$$\begin{aligned} c_t(\bar{\boldsymbol{\xi}}_t^{\mathcal{I}}, \boldsymbol{u}_t) &= \sum_{t=t_0}^{T} (\bar{\boldsymbol{\xi}}_t^{\mathcal{I}} - \bar{\boldsymbol{\mu}}_t)^\top \boldsymbol{Q}_t (\bar{\boldsymbol{\xi}}_t^{\mathcal{I}} - \bar{\boldsymbol{\mu}}_t) + \boldsymbol{u}_t^\top \boldsymbol{R}_t \boldsymbol{u}_t, \\ &\text{s.t.} \quad \bar{\boldsymbol{\xi}}_{t+1}^{\mathcal{I}} = \boldsymbol{A}_d \bar{\boldsymbol{\xi}}_t^{\mathcal{I}} + \boldsymbol{B}_d \boldsymbol{u}_t, \end{aligned}$$

starting from the initial value $\bar{\boldsymbol{\xi}}_{t_0}^{\mathcal{I}} = \begin{bmatrix} \boldsymbol{\xi}_{t_0}^{\mathcal{I}\top} & \boldsymbol{0}^\top \end{bmatrix}^\top$, with $\bar{\boldsymbol{\xi}}_t^{\mathcal{I}} = \begin{bmatrix} \boldsymbol{\xi}_t^{\mathcal{I}\top} & \boldsymbol{\xi}_{t+1}^{\mathcal{I}\top} \end{bmatrix}^\top$, $\bar{\boldsymbol{\mu}}_t = \begin{bmatrix} \hat{\boldsymbol{\mu}}_t^\top & \boldsymbol{0}^\top \end{bmatrix}^\top$, $\boldsymbol{A}_d = \begin{bmatrix} \boldsymbol{I} & \Delta t \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}$ and $B_d = \begin{bmatrix} \frac{1}{2}\Delta t^2 \boldsymbol{I} \\ \Delta t \boldsymbol{I} \end{bmatrix}$. Solving the dynamic Riccati equation backwards in time gives the optimal control input $\boldsymbol{u}_t^* \in \mathbb{R}^7$. For the infinite horizon case in shared control with $\boldsymbol{Q}_t = \boldsymbol{Q}, T \to \infty$ and the desired pose $\hat{\boldsymbol{\mu}}_t = \hat{\boldsymbol{\mu}}_{t_0}$, the control law is obtained by eigendecomposition of the discrete algebraic Riccati equation. The resulting path $\boldsymbol{\xi}_t^{*\mathcal{I}}$ smoothly follows the desired pose/trajectory $\hat{\boldsymbol{\mu}}_t$ and the computed gains stabilize $\boldsymbol{\xi}_t^{\mathcal{I}}$ along $\boldsymbol{\xi}_t^{*\mathcal{I}}$ in accordance with the precision required during the task. The tracking force on the remote arm $\boldsymbol{F}_r \in \mathbb{R}^6$ is computed with a proportional-derivative (PD) controller

using fixed pose $\boldsymbol{K}_r^p \in \mathbb{R}^{6\times 7}$ and twist gains $\boldsymbol{K}_r^v \in \mathbb{R}^{6\times 7}$,

$$\begin{aligned} \boldsymbol{F}_r &= \boldsymbol{K}_r^p(\boldsymbol{\xi}_t^{*\mathcal{I}} + \boldsymbol{\xi}_{\text{offset}} - \boldsymbol{\xi}_r) - \boldsymbol{K}_r^v \dot{\boldsymbol{\xi}}_r, \\ \boldsymbol{\tau}_{q_r} &= \boldsymbol{J}_{q_r}^\top \boldsymbol{F}_r + \left(\boldsymbol{I} - \boldsymbol{J}_{q_r}^\top \boldsymbol{J}_{q_r}^{\dagger\,\top}\right) k_{q_p}(\boldsymbol{q}_{\text{neu}} - \boldsymbol{q}_r) - k_{q_v}\dot{\boldsymbol{q}}_r, \end{aligned} \tag{6.11}$$

where $\boldsymbol{\xi}_{\text{offset}}$ maps the desired pose in the workspace of the remote arm, $\boldsymbol{\xi}_r, \dot{\boldsymbol{\xi}}_r \in \mathbb{R}^7$ is the pose and twist of the end-effector of the remote arm, $\boldsymbol{J}_{q_r} \in \mathbb{R}^{6\times 7}$ is the Jacobian of the remote arm that maps the tracking force $\boldsymbol{F}_r$ to the joint torques $\boldsymbol{\tau}_{q_r} \in \mathbb{R}^7$. $(\boldsymbol{I} - \boldsymbol{J}_{q_r}^\top \boldsymbol{J}_{q_r}^{\dagger\,\top})$ is the null space of the Jacobian with right pseudoinverse $\boldsymbol{J}_{q_r}^\dagger = \boldsymbol{J}_{q_r}^\top (\boldsymbol{J}_{q_r}\boldsymbol{J}_{q_r}^\top)^{-1}$, that keeps the joint configuration of the remote arm $\boldsymbol{q}_r \in \mathbb{R}^7$ similar to the neutral starting position $\boldsymbol{q}_{\text{neu}} \in \mathbb{R}^7$ with tracking gain k_{q_p}. The last term $k_{q_v}\dot{\boldsymbol{q}}_r$ dampens the joint velocities of the remote arm by gain k_{q_v}. The two arms are clutched during teleoperation, and the remote arm is teleoperated under unilateral control mode, i.e., no force is fed back to the teleoperator. Using a haptic interface to feed back interaction forces on the teleoperation site is subject to future work.

6.4 Comparison with Virtual Fixtures

Virtual fixtures or virtual guides are used to constrain the movement of the remote arm to follow a desired trajectory [Rosenberg 1993, Abbott 2007]. In [Raiola 2015], the end-effector of the teleoperator arm is virtually coupled to the desired trajectory by a spring damper system. Like a cart being pulled on a rail, the teleoperator arm movement induces the motion of the remote arm along the trajectory (see Fig. 6.6 for the use of virtual guided fixtures in teleoperation). The desired remote arm pose along the trajectory $\hat{\boldsymbol{\mu}}_{s_{vm}}$ is specified by the phase variable s_{vm} with $s_{vm} = 0$ at the beginning of the trajectory, $s_{vm} = 1$ at the end of the trajectory, and $\dot{\hat{\boldsymbol{\mu}}}_{s_{vm}} = \boldsymbol{J}_{s_{vm}}\dot{s}_{vm}$ where $\boldsymbol{J}_{s_{vm}} \in \mathbb{R}^7$ is the mapping Jacobian. The teleoperator arm movement $\boldsymbol{\xi}_t^{\mathcal{I}}$ induces the dynamics on the phase variable with

$$\dot{s}_{vm} = \left(\boldsymbol{J}_{s_{vm}}^\top \boldsymbol{B}_\sigma \boldsymbol{J}_{s_{vm}}\right)^{-1} \boldsymbol{J}_{s_{vm}}^\top \left(\boldsymbol{K}_\sigma(\boldsymbol{\xi}_t^{\mathcal{I}} - \hat{\boldsymbol{\mu}}_{s_{vm}}) + \boldsymbol{B}_\sigma \dot{\boldsymbol{\xi}}_t^{\mathcal{I}}\right), \tag{6.12}$$

where $\boldsymbol{K}_\sigma$ and $\boldsymbol{B}_\sigma$ define the stiffness and the damping of the virtual fixture.

A task-parameterized HSMM can be used as a virtual fixture by augmenting the teleoperator data with the phase variable s_{vm} during the demonstrations step. In the teleoperation phase, the desired remote arm pose is retrieved by Gaussian conditioning on the phase variable (see Eq. (6.5)) with $\mathcal{P}(\hat{\boldsymbol{\mu}}_{s_{vm}}|s_{vm}) \approx \mathcal{N}(\tilde{\boldsymbol{\mu}}_t^{\mathcal{O}}, \tilde{\boldsymbol{\Sigma}}_t^{\mathcal{O}})$ while the Jacobian $\boldsymbol{J}_{s_{vm}}$ is obtained by evaluating the analytical derivative of Eq. (6.5) with respect to s_{vm}. Note that

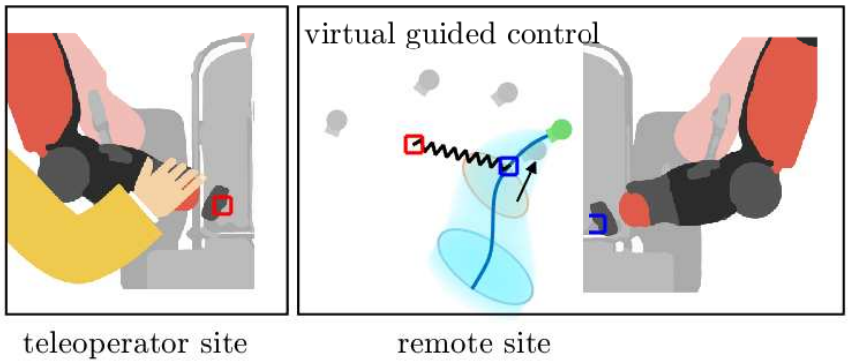


Figure 6.6: Virtual guided fixtures for teleoperation. The teleoperator end-effector (in red square) is virtually connected to the remote arm end-effector (in blue square) with a spring-damper system. The movement of the teleoperator arm guides the remote arm along the desired trajectory. The desired trajectory is adapted locally according to the remote situation.

the input component here is s_{vm} and the output component $\mathcal{N}(\tilde{\boldsymbol{\mu}}_t^{\mathcal{O}}, \tilde{\boldsymbol{\Sigma}}_t^{\mathcal{O}})$ gives the desired pose $\hat{\boldsymbol{\mu}}_{s_{vm}}$ and its uncertainty, along with the Jacobian J_{vm} that governs the evolution of the phase variable in Eq. (6.12) to guide the arm along the trajectory.

In virtual fixture control, the teleoperator arm movement governs the evolution of the phase variable and Gaussian conditioning on the phase variable gives the desired pose of the remote arm; whereas in shared control, Gaussian conditioning on the teleoperator arm pose gives the desired pose of the remote arm. In our implementation of virtual guides, we used the logistic function for the phase variable (instead of the linear ramp function) to slow down the cart at the beginning and at the end of the trajectory. The transformation of the phase variable is important to limit injection of arbitrarily high velocities in Eq. (6.12). Alternatively, the trajectory length can also be used as an input variable to normalize the Jacobian and control the velocities of the cart in the boundary conditions [Raiola 2015]. Moreover, more datapoints are typically required than in shared control to ensure smooth evolution of the phase variable during teleoperation.

6.5 Experiments, Results and Discussions

In this section, we evaluate our semi-autonomous teleoperation framework for reaching a movable screwdriver target (see Sec. 5.7.2) and opening a valve (see Sec. 3.4) with the Baxter robot. Our experimental protocol remains the same as described in Sec. 6.1.

Reaching a Movable Object: The objective of this task is to reach a target point

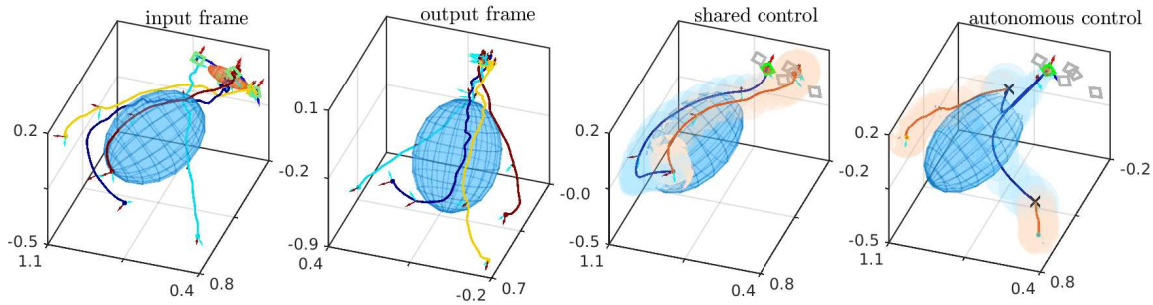


Figure 6.7: Reaching a movable target (in green squares) for screw driving: *(left)* demonstrations and model shown in the input frame; *(center-left)* demonstrations and model in the output frame; *(center-right)* the teleoperator demonstration (in red) is corrected under shared control (in blue) to reach a new target location shown in green; *(right)* the teleoperator switches from direct control to autonomous control mode (marked with a cross) after which the movement is autonomously generated to the new target location.

with a screwdriver while adapting the movement for different target configurations. We describe the task with a single frame $\{\boldsymbol{A}_1, \boldsymbol{b}_1\}$ attached to the target and collect 6 kinesthetic demonstrations (4 for training and 2 for testing) with the initial pose of the target rotated/translated in the successive demonstrations. Results of the joint distribution with 2 mixture components for different target poses are shown in Fig. 6.7. Demonstrations for the input frame represent the movement of the end-effector of the teleoperator's arm to different target poses, while the output frame maps the demonstrations to a pose as observed from the target perspective.

Opening a Valve: The goal of this task is to bring the valve in an open position from different initial configurations of the valve [Tanwani 2016a]. The task is described by two frames, one with the observed initial configuration of valve $\{\boldsymbol{A}_1, \boldsymbol{b}_1\}$ and the other with the desired end configuration of the valve $\{\boldsymbol{A}_2, \boldsymbol{b}_2\}$. The changing configuration of the valve is tracked using an augmented reality (AR) tag with a Kinect 2.0. We record 8 kinesthetic demonstrations (6 for training and 2 for testing) with the initial configuration of the valve corresponding to $\{180, 135, 90, 45, 157.5, 112.5, 67.5, 22.5\}$ degrees with the horizontal in the successive demonstrations. Results of the learned model with 2 frames and 7 mixture components are shown in Fig. 6.8. The input components of both frames represent the demonstrations identically in global coordinates. The output components of the frames depict high variability in reaching the valve and coming back to the home position, whereas there is no variation in grasping/turning and stopping the valve in their respective coordinate systems. This allows the robot arm to reach the valve from different configurations, grasp the valve and turn it to the desired open position.

All the demonstrations are collected with a controller compensating for the effect of gravity

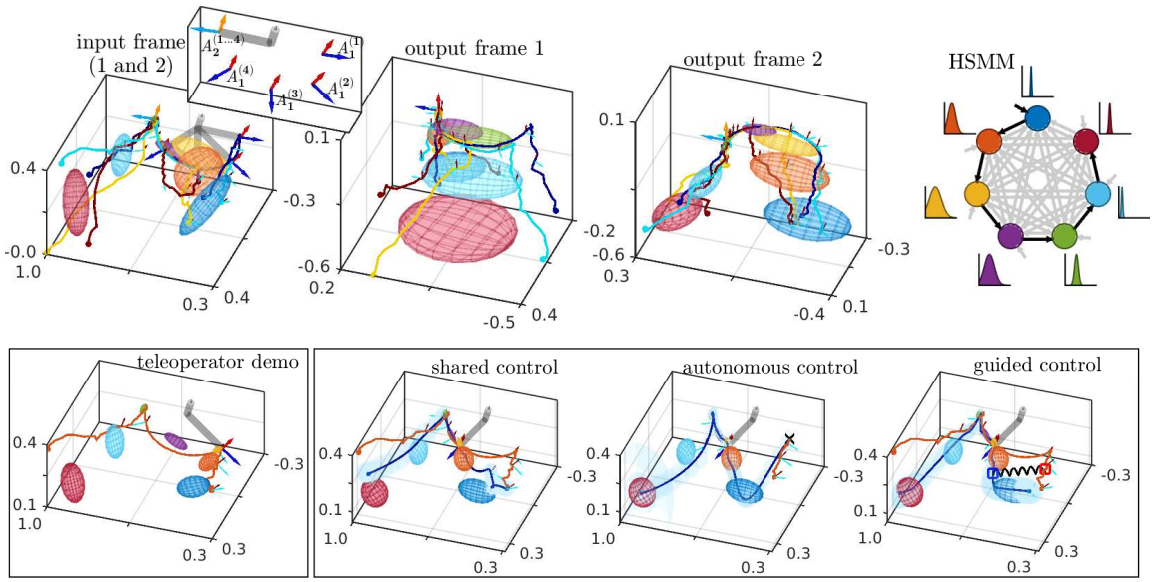


Figure 6.8: Open valve (in gray) from different initial configurations $\boldsymbol{A}_1^{(i)}$ to final configuration $\boldsymbol{A}_2$: *(top)* learned model in the input and the output frames, and left-right HSMM with state transition and state duration model ($s^{\max}=100$); *(bottom)* the teleoperator performs the task (in red) with respect to the perceived valve configuration on the left, where the different control modes assist the remote arm (in blue) to perform the task with actual valve configuration. The spring displayed in bottom right inset represents the virtual fixture between the teleoperator pose and the desired pose along the trajectory.

by a human operator who is familiar with the robot, but not an expert in teleoperation. We evaluate the performance of our approach using three different criteria: 1) task performance error, 2) environmental differences, and 3) execution time.

6.5.1 Task Performance Error

Our objective is to assist the teleoperator to perform remote manipulation tasks in a repeatable and precise manner while reducing the workload of the teleoperator. Results of the shared and autonomous modes of assistance for target reaching task are shown in Fig. 6.7. In shared control, the model corrects the movement of the teleoperator in accordance with the output component of the model that adapts locally to the target. If the variance of the resulting conditional distribution is low, the correction is stronger and vice versa. While demonstrating autonomous control, the teleoperator tests the system by randomly switching between control modes during the task. Fig. 6.7 (*right*) shows how the movement of the robot converges to the target from different switching instants, while being repeatable and more precise than the direct and the shared control results.

Table 6.1: Performance comparison of the teleoperation modes with direct control (DC), shared control (SC), autonomous control (AC), and virtual fixture control (VFC). N_p is the number of parameters of the model, and the errors represent average mean squared errors between the demonstrations and the model predictions (in centimeters).

SC		**AC**		**VFC**	
Train	**Test**	**Train**	**Test**	**Train**	**Test**
valve opening ($K = 7$, **DC Train** $= 1.412 \pm 1.20$, **DC Test** $= 1.261 \pm 1.13$)					
0.717 ± 0.67	0.721 ± 0.68	0.737 ± 0.62	0.464 ± 0.33	2.836 ± 2.07	2.011 ± 0.83
target snapping ($K = 2$, **DC Train** $= 1.954 \pm 2.04$, **DC Test** $= 1.959 \pm 1.81$)					
0.23 ± 0.25	0.31 ± 0.26	0.109 ± 0.08	0.178 ± 0.09	0.183 ± 0.12	0.311 ± 0.27

6.5.2 Robustness to Different Environments

Performance of the teleoperator is typically affected by the environmental differences between the teleoperator and the remote sites. Such differences exist as streaming full OctoMaps over satellite communication for updating the remote environment on the teleoperator site are only possible at a very low frequency. In Fig. 6.8, we compare different assistance approaches to handle these discrepancies by setting different configurations of the valve on the teleoperator and the remote end. We can see that the task-parameterized model successfully adapts to the external situation on the teleoperator and the remote site, thereby, mitigating the difference of environmental situation on the two sites. The teleoperator performs the movement according to the perceived valve configuration or switches to the autonomous mode while performing the task, and the generated movement is adapted locally with shared, autonomous or virtual fixture control.

Table 6.1 summarizes the results of different control modes to mitigate imprecise teleoperator movements with our model. For each target or valve configuration in the training or testing set, all the demonstrations in the training and testing sets are treated in a given control mode and compared with the human demonstration for that particular target. Mean-squared endpoint errors for target tracking and mean-squared trajectory errors for valve opening tasks are averaged over all demonstrations and for all target or valve configurations. The results show that the autonomous control gives the most repeatable and precise assistance among different teleoperation modes. Moreover, we observe high performance errors of the virtual fixture control on valve opening task as change of movement directions in the teleoperator demonstration tends to induce remote arm movement in the reverse direction along the trajectory, often resulting in an unsuccessful trial.

Table 6.2: Average execution time of performing a manipulation task in seconds under different teleoperation modes.

Task	DC	SC	AC	VFC	Preferred Mode
valve opening	18.2	13.6	**12.3**	13.1	**AC**
target snapping	7.2	**4.3**	5.1	5.6	**SC**

6.5.3 Execution Time

In order to evaluate the effect of teleoperation mode on the task execution time, the human operator performs the task 5 times for each teleoperation mode from different initial conditions. At the end of all trials, the operator reveals the preferred mode of assistance for each task. Results in Table 6.2 suggest performance improvement in task execution time using the learned model as compared to the direct teleoperation mode.

6.6 Conclusion

In this chapter, we used the task-parameterized HSMMs for semi-autonomous teleoperation of remote manipulation tasks. The HSMM clustered the demonstrations into meaningful segments/primitives and encoded the transition patterns among the segments. Using the methodology, we presented our approach to assist the teleoperator using the learned model by: 1) continuously correcting the movement of the remote arm given the teleoperator arm data based on shared control, or 2) generating the movement of the remote arm based on autonomous control. We compared our approach with virtual fixtures to benchmark the manipulation assistance methods in teleoperation. We established the merits of our approach by: 1) allowing the teleoperator to perform the task locally with respect to the perceived environment(often delayed/inconsistent compared to the actual remote environment), and adapting the movement locally with the actual situation using the task-parameterized formulation, 2) improving the task performance of the teleoperator by mitigating the effect of imprecise movements using time-independent shared control and/or time-dependent autonomous control, and 3) reducing the average execution time of performing a remote manipulation task.

Chapter 7

Concluding Remarks and Future Directions

Robotics is entering in a golden age where machine learning is becoming well poised to tackle large scale real world problems. In this thesis, we have explored several frontiers in imitation and reinforcement learning for acquiring manipulation skills in robots. We summarize the findings of the thesis in this chapter and provide an outlook to the future directions.

- **Is reinforcement learning from scratch practical for skill acquisition in real world ?**

Our work on actor-critics with experience replay for model-free reinforcement learning shows high sample complexity for simulated running of half-cheetah and reaching of octopus arm tasks [Wawrzynski 2013]. Learning from scratch on real world tasks in a model-free manner is often dangerous due to the explicit noise added during exploration, along with the high sample and time complexity required to find a reasonable control policy. Model-based methods avoid the problem of explicit exploration by improving the models estimate for a given policy, but often require good initial model to start learning. Using human demonstrations to seed the initial policy/value-function/dynamics model gives promising results for a variety of tasks [Tanwani 2013a, Tanwani 2014]. Note that the deep learning variants of these algorithms are also gaining popularity for learning directly from images [Mnih 2013] and for bridging the gap between simulation and reality for skill acquisition [Rusu 2016].

- **How is inverse reinforcement learning useful in learning from demonstrations ? Does learning multiple reward functions help in learning better/richer control policies ?**

Inverse RL aims to recover the unknown reward function that is being optimized in the human demonstrations. It remains an ill-posed problem as many reward functions are optimal for a given set of demonstrations. This requires a lot of engineering effort in setting up the features of the reward function as there is no direct correspondence between the reward function features and the policy samples. Although some effort has been employed in learning reward function directly from images and/or reinterpreting the reward function as a sub-goal or a trajectory, solving the reward function parameters in its original formulation remains a difficult problem at this stage for learning new skills where the dynamics model, reward function and optimal policy are all unknown. To mitigate the ill-posed nature of the problem, our efforts have been on learning multiple reward functions that can instead encapsulate a set of useful behaviours in the demonstrations while using transfer of knowledge to bootstrap the learning process [Tanwani 2013b]. This primarily relaxes the strict assumption of demonstrations being optimal with respect to a particular reward function and allows the demonstrations to be unstructured by being optimal with respect to multiple reward functions.

- **What are the promising ways of teaching new skills to robots from human demonstrations ?**

Considering the sample and time complexity of (inverse) RL to learn new manipulation tasks, robot learning directly from human demonstrations is a promising way to acquire new skills. Most of this work has been on direct trajectory learning in a supervised manner using DMPs or dynamical systems in general. In this thesis, we have emphasized learning manipulation skills from human demonstrations in an unsupervised manner using a family of **hidden Markov models** by sequencing the atomic movement segments or primitives [Tanwani 2016a, Tanwani 2016c]. Recently, several other approaches are being proposed for deep imitation learning from images directly such as Generative Adversarial Imitation Learning [Ho 2016], one-shot imitation learning [Duan 2017, Finn 2017], and time contrastive networks [Sermanet 2017]. Further work will be required to combine these different learning strategies and determine in which settings they can be applied.

- **Why are generative models useful in learning robot manipulation skills ?**

Generative models typically learn the probability density function of the demonstrations

in order to regenerate new samples from the model. This is in contrast to discriminative models that would typically model the target variables directly by regression. While discriminative training has arguably been performing better over the years, the generative models encapsulate complex relationships between target and observed variables and provide a direct way to evaluate the model with the regenerated samples. Common generative models include Gaussian mixture models, hidden Markov models, latent Dirichlet allocation, variational autoencoders, restricted Boltzmann machines, and generative adversarial networks. In this thesis, we have built upon a family of hidden Markov models, namely **hidden semi-Markov models**, for acquiring manipulation tasks from human demonstrations. Hidden semi-Markov models posit a duration interval for each state/action, thereby, segmenting the demonstrations into sub-goals that are sequenced together in performing manipulation tasks [Tanwani 2016a]. We have presented several of its variants in this thesis and demonstrated its use for learning robot manipulation tasks from a few demonstrations with no labeled examples. In comparison to inverse reinforcement learning approaches, the models are straightforward to use and require considerably less time in training and validating the model.

- **What is the role of task-parameterized models? How did the task-parmeterized formulations of the hidden semi-Markov models help in learning robot manipulation tasks ?**

Task-parameterized models encapsulate the invariant representations of the task by adapting the model parameters with respect to the changing task parameters describing the environmental situation. The task parameters refer to the coordinate systems that, for example, can move with the position/orientation of objects, or scale with the size of the objects of interest in the environment. Capturing such invariant representations has allowed us to compactly encode the task variations than using a standard regression problem. We have presented their formulation and applications with hidden semi-Markov model and its variants for latent space and Bayesian non-parametric online learning of robot manipulation skills [Tanwani 2016a, Tanwani 2016c, Tanwani 2017]. An important direction of future work is to not rely on specifying the task parameters manually, but to infer them simultaneously from demonstrations.

- **How did these task-parameterized hidden semi-Markov models scale with high-dimensional data and limited number of demonstrations ?**

Modeling probabilistic distributions in high-dimensional spaces is a challenging problem. We advocate learning in latent spaces to exploit the structure of the problem in order

to overcome this problem with generative models. In this thesis, we have widely exploited two basic latent space representations in our formulations: 1) Mixture of factor Analyzers (MFA) or mixture of probabilistic principal component analysis (MPPCA), 2) semi-tied model parameters. MFA/MPPCA gives a separate low-rank decomposition of each covariance matrix in the mixture model [Tanwani 2016b, Tanwani 2016c]; whereas the semi-tied representation shares a set of basis vectors across all the mixture components [Tanwani 2016a]. The latent space representations allow the model to generalize better and exploit coordination patterns with noisy and/or insufficient training data. Our experiments yield comparable or better performance in latent space representations with much less parameters than mixture models with full covariance matrices. A systematic investigation of the effect of number of basis vectors in these latent space models is a promising direction of future work (see extended maximum likelihood linear transform (EMLLT) [Olsen 2004], and multiple linear transforms (MLT) [Goel 2001] for reference). Our other promising direction of future work is to exploit the structural constraints of the data (quaternions, tensors) and/or the model parameters (symmetry, orthogonality, low-rank) in learning the mixture models (see [Jaquier 2017] for example).

- **How did the Bayesian non-parametric formulations of the model perform in learning manipulation skills online from human demonstrations ?**

Bayesian non-parametric mixture models avoid problem in model selection which is required for online learning problems. Computational overhead of existing sampling-based and variational techniques led us to present online formulations of Bayesian non-parametric mixture models under small variance asymptotics [Tanwani 2016b]. The resulting scalable online sequence clustering algorithm incrementally clusters the streaming data with non-parametric locally linear Dirichlet process MPPCA and encodes the temporal information in an infinite hierarchical Dirichlet process hidden semi-Markov model. The non-parametric skill encoding scheme can encode a wide range of tasks, while being robust to environmental situations with the task-parameterized formulation [Tanwani 2016c]. In future work, we plan to bootstrap the online learning process with the batch algorithm after a few initial demonstrations of the task. We would like to use the initialized model to make an educated guess about the penalty parameters for non-parametric online learning.

- **Teleoperation vs shared autonomy vs full autonomy? How does the role of human operator vary in performing manipulation tasks ?**

Human-in-the-loop robot learning is going to speed up the process of making robots an everyday reality. On one end of this spectrum, we have direct teleoperation where a human

operator directly commands the robot to follow in joint space/task space in performing an underlying task, and on the other hand of the spectrum is the full autonomy where the robot gathers the intent of the teleoperator and autonomously executes the task without any human intervention [Tanwani 2017]. The shared autonomy lies in between the two ends and provides manipulation assistance as a virtual guide to the teleoperator from the learned model of the task. Semi-autonomy instead of full autonomy is desirable where full autonomy is too hard to provide satisfying results (e.g., driving assistance vs self-driving cars). Semi-autonomous manipulation allows the operator to reach distant hazardous environments and gain confidence in controlling the system, which is a potential barrier in several real world applications.

- **How feasible is it to perform remote manipulation tasks over satellite communication in the presence of delays? How the generative models helped to perform remote manipulation tasks in the context of DexROV ?**

Performing remote manipulation tasks over satellite communication is difficult because of the delays introduced in the transmission and the feedback of the signals. In naive implementation, the teleoperator typically has to resort to a piecewise move-and-wait strategy in order to wait for the remote arm to catch up with the teleoperator. Within DexROV, we have designed and developed a cognitive engine to mitigate the effect of delays in performing remote manipulation tasks. The cognitive engine – comprised of the task-parameterized generative models – recognizes the intention of the teloperator on one hand and adjusts the movement of the robot arm to assist the teleoperator in performing remote manipulation tasks using the same model. We have seen how the shared and autonomous control modes of the cognitive engine reduce task errors and the execution time, while handling differences of contexts between the teleoperator site (virtual reality environment) and the remotely operated vehicle site [Tanwani 2017]. In our future work, we plan to test the models with large communication delays over satellite communication in real underwater environments.

- **Where does the field of robot learning stand today ? What are the next milestones in robot learning from demonstrations ?**

Robot learning is increasingly gaining more interest from various scientific communities including neuroscience, machine learning, artificial intelligence, computer vision, control systems and human-computer interaction. There is a global consensus that robots need to learn and adapt their skills to pave their way into the real world. The main prospective

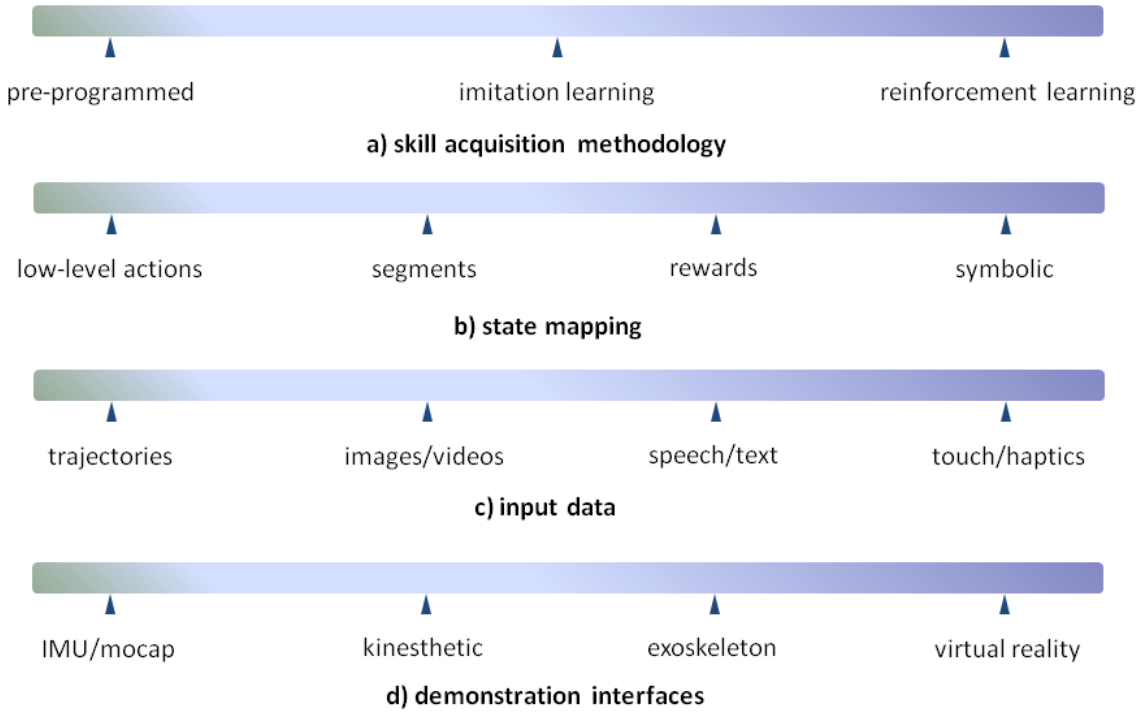


Figure 7.1: Spectrum of related robot learning directions.

directions of research and development in robot learning are somewhat varied depending upon the scientific community and the application scenario. In Fig. 7.1, we have outlined a broad spectrum of research directions related to our work in this thesis. The discretion is made on the basis of: 1) skill acquisition methodology – whether the skills are pre-programmed, learned from human demonstrations and/or acquired with reinforcement learning; 2) state mapping – whether the state of the environment is mapped to low-level actions, segments/sub-goals, rewards or a high-level symbolic planner; 3) input data – whether the state of environment is described by trajectories/poses, images/videos, speech/text and/or haptic feedback; 4) demonstration interfaces – what interfaces are used for collecting data from user such as IMU, motion capture markers (mocap), kinesthetic demonstrations or using wearables such as exoskeleton and virtual reality headsets. A general trend of advancement with deep learning techniques has been to encompass the right end of the spectrum with abstract state representations, multiple interface modalities, and using high performance computing resources to deal with large scale sensory data. Nevertheless, the real world applications of robotics today are more concentrated on the left end of the spectrum. One of the main challenges to resolve going forward is to standardize the datasets, algorithms, architectures, simulators, low-level/high-level primitives and the hardware platforms. Having a library of standardized segments such that the appropriate segments can be sequenced together in a given situation is a powerful way of performing

complex manipulation tasks. This will help to plan over much longer time horizons than managed with current robot learning methodologies. Unsupervised learning is well-suited for dealing with copious amounts of unlabelled sensory data in future. Although, it seems not clear at this stage what internal predictive representations need to be acquired of the objects (and their relationships with other objects) in the environment so that the robots can learn and reason about what to do in a given situation in order to interact with humans in everyday life tasks.

The main take away message of the thesis is that robot learning from humans is a promising way to acquire new skills. Generative mixture models are useful for learning from a few examples that are not explicitly labelled. The contributions are inspired by the need to make generative mixture models easy to use for robot learning in a variety of applications, while requiring considerably less time in implementation. We have presented formulations for learning invariant task representations with hidden semi-Markov models for recognition, prediction, and reproduction of manipulation tasks; learning in latent space for robust parameter estimation of mixture models with high-dimensional data; and learning online with Bayesian non-parametric mixture models under small variance asymptotics for streaming data. The cognitive engine based on these generative mixture models is used for recognizing the intention of the user and providing assistance to remotely perform manipulation skills by semi-autonomous teleoperation. As a result, the operation costs of teleoperating dexterous remotely operated vehicles are reduced and the efficiency of the teleoperator is improved in performing remote manipulation tasks.

Bibliography

[Aarno 2008] Daniel Aarno and Danica Kragic. *Motion Intention Recognition in Robot Assisted Applications*. Robot. Auton. Syst., vol. 56, no. 8, pages 692–705, August 2008.

[Abbeel 2004] Pieter Abbeel and Andrew Y. Ng. *Apprenticeship Learning via Inverse Reinforcement Learning*. In Proceedings of the Twenty-first International Conference on Machine Learning. ACM Press, 2004.

[Abbeel 2010] Pieter Abbeel, Adam Coates and Andrew Y Ng. *Autonomous Helicopter Aerobatics through Apprenticeship Learning*. The International Journal of Robotics Research, vol. 29, no. 13, pages 1608–1639, 2010.

[Abbott 2007] Jake J. Abbott, Panadda Marayong and Allison M. Okamura. Haptic virtual fixtures for robot-assisted manipulation, pages 49–64. 2007.

[Abi-Farraj 2017] Firas Abi-Farraj, Takayuki Osa, Nicoló Pedemonte, Jan Peters, Gerhard Neumann and Paolo Giordano Robuffo. *A Learning-based Shared Control Architecture for Interactive Task Execution*. In Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), pages 329–335, Singapore, 2017.

[Aghasadeghi 2011] Navid Aghasadeghi and Timothy Bretl. *Maximum entropy inverse reinforcement learning in continuous state spaces with path integrals*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1561–1566, 2011.

[Alissandrakis 2006] A. Alissandrakis, C. L. Nehaniv and K. Dautenhahn. *Action, State and Effect Metrics for Robot Imitation*. In Proc. IEEE Intl Symp. on Robot and Human Interactive Communication (Ro-Man), pages 232–237, Hatfield, UK, September 2006.

[Antonelli 2015] G. Antonelli, S. Moe and K. Y. Pettersen. *Incorporating set-based control within the singularity-robust multiple task-priority inverse kinematics*. In 2015 23rd

Mediterranean Conference on Control and Automation (MED), pages 1092–1097, 2015.

[Argall 2009] Brenna D. Argall, Sonia Chernova, Manuela Veloso and Brett Browning. *A Survey of Robot Learning from Demonstration.* Robot. Auton. Syst., vol. 57, no. 5, pages 469–483, May 2009.

[Asfour 2008] Tamim Asfour, Pedram Azad, Florian Gyarfas and Rüdiger Dillmann. *Imitation Learning of Dual-Arm Manipulation Tasks in Humanoid Robots.* I. J. Humanoid Robotics, vol. 5, no. 2, pages 183–202, 2008.

[Atkeson 1997a] C. G. Atkeson and S. Schaal. *robot learning from demonstration.* In Proceedings of the fourteenth international conference, pages 12–20. morgan kaufmann, 1997.

[Atkeson 1997b] Christopher G. Atkeson, Andrew W. Moore and Stefan Schaal. *Locally Weighted Learning.* Artif. Intell. Rev., vol. 11, no. 1-5, pages 11–73, February 1997.

[Babes 2011] Monica Babes, Vukosi Marivate, Michael Littman and Kaushik Subramanian. *Apprenticeship Learning About Multiple Intentions.* In Proceedings of the 28th International Conference on Machine Learning, ICML, pages 897–904. ACM, 2011.

[Beal 2002] Matthew J. Beal, Zoubin Ghahramani and Carl E. Rasmussen. *The Infinite Hidden Markov Model.* In Machine Learning, pages 29–245, 2002.

[Bellas 2013] Anastasios Bellas, Charles Bouveyron, Marie Cottrell and Jérôme Lacaille. *Model-based clustering of high-dimensional data streams with online mixture of probabilistic PCA.* Advances in Data Analysis and Classification, vol. 7, no. 3, pages 281–300, 2013.

[Berio 2017] D Berio, S. Calinon and F. Fol Leymarie. *Dynamic Graffiti Stylisation with Stochastic Optimal Control.* In ACM Proceedings of the 4th International Conference on Movement and Computing, MOCO '17. ACM, 2017.

[Bertrand 1985] O Bertrand, F Perrin and J Pernier. *A theoretical justification of the average reference in topographic evoked potential studies.* Electroencephalography and Clinical Neurophysiology/Evoked Potentials Section, vol. 62, no. 6, pages 462–464, 1985.

[Bertsekas 2012] D.P. Bertsekas. Dynamic programming and optimal control. Athena Scientific, 2012.

[Bhatnagar 2009] S. Bhatnagar, R. Sutton, M. Ghavamzadeh and M. Lee. *Natural Actor-Critic Algorithms.* Automatica, vol. 45, pages 2471–2482, 2009.

[Billard 2016] A. G. Billard, S. Calinon and R. Dillmann. *Learning From Humans*. In B. Siciliano and O. Khatib, editors, Handbook of Robotics, chapter 74, pages 1995–2014. Springer, Secaucus, NJ, USA, 2016. 2nd Edition.

[Borrelli 2011] Francesco Borrelli, Alberto Bemporad and Manfred Morari. Predictive control for linear and hybrid systems. Cambridge University Press, 2011.

[Boularias 2011] Abdeslam Boularias, Jens Kober and Jan Peters. *Relative entropy inverse reinforcement learning*. Artificial Intelligence and Statistics (AISTATS 2011), vol. 15, pages 20–27, 2011.

[Bouveyron 2007] C. Bouveyron, S. Girard and C. Schmid. *High-dimensional data clustering*. Computational Statistics and Data Analysis, vol. 52, no. 1, pages 502–519, 2007.

[Bouveyron 2014] C. Bouveyron and C. Brunet. *Model-based clustering of high-dimensional data: A review*. Computational Statistics and Data Analysis, vol. 71, pages 52–78, 2014.

[Broderick 2013] Tamara Broderick, Brian Kulis and Michael I. Jordan. *MAD-Bayes: MAP-based Asymptotic Derivations from Bayes*. In Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013, pages 226–234, 2013.

[Bruno 2016] Danilo Bruno, Sylvain Calinon and Darwin G. Caldwell. *Learning autonomous behaviours for the body of a flexible surgical robot*. Autonomous Robots, pages 1–15, 2016.

[Calinon 2010] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell and A. G. Billard. *Learning and reproduction of gestures by imitation: An approach based on Hidden Markov Model and Gaussian Mixture Regression*. IEEE Robotics and Automation Magazine, vol. 17, no. 2, pages 44–54, 2010.

[Calinon 2011] S. Calinon, A. Pistillo and D. G. Caldwell. *Encoding the time and space constraints of a task in explicit-duration hidden Markov model*. In Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), pages 3413–3418, San Francisco, CA, USA, September 2011.

[Calinon 2016] S. Calinon. *A Tutorial on Task-Parameterized Movement Learning and Retrieval*. Intelligent Service Robotics, vol. 9, no. 1, pages 1–29, 2016.

[Campbell 2013] Trevor Campbell, Miao Liu, Brian Kulis, Jonathan P. How and Lawrence Carin. *Dynamic Clustering via Asymptotics of the Dependent Dirichlet Process Mix-*

ture. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani and Kilian Q. Weinberger, editors, NIPS, pages 449–457, 2013.

[Chen 2010] Minhua Chen, Jorge G. Silva, John William Paisley, Chunping Wang, David B. Dunson and Lawrence Carin. *Compressive Sensing on Manifolds Using a Nonparametric Mixture of Factor Analyzers: Algorithm and Performance Bounds.* IEEE Trans. Signal Processing, vol. 58, no. 12, pages 6140–6155, 2010.

[Choi 2012] Jaedeug Choi and Kee-Eung Kim. *Nonparametric Bayesian Inverse Reinforcement Learning for Multiple Reward Functions.* In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou and K.Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 314–322. 2012.

[Cichosz 1999] Pawel Cichosz. *An analysis of experience replay in temporal difference learning.* Cybernetics and Systems, vol. 30, pages 341–363, 1999.

[Coates 2008] Adam Coates, Pieter Abbeel and Andrew Y. Ng. *Learning for control from multiple demonstrations.* In Proceedings of the 25th international conference on Machine learning, ICML, pages 144–151. ACM, 2008.

[d'Avella 2003] A. d'Avella, P. Saltiel and E. Bizzi. *Combinations of muscle synergies in the construction of a natural motor behavior.* Nature Neuroscience, vol. 6, pages 300–308, 2003.

[Dayan 1997] Peter Dayan and Geoffrey E. Hinton. *Using expectation-maximization for reinforcement learning.* Neural Computation, vol. 9, pages 271–278, 1997.

[Deisenroth 2013] Marc Peter Deisenroth, Gerhard Neumann and Jan Peters. *A Survey on Policy Search for Robotics.* Foundations and Trends® in Robotics, vol. 2, no. 1–2, pages 1–142, 2013.

[Dempster 1977] A. P. Dempster, N. M. Laird and D. B. Rubin. *Maximum Likelihood from Incomplete Data via the EM Algorithm.* Journal of the Royal Statistical Society B, vol. 39, no. 1, pages 1–38, 1977.

[Dimitrakakis 2012] Christos Dimitrakakis and Constantin A. Rothkopf. *Bayesian multitask inverse reinforcement learning.* In Proceedings of the 9th European conference on Recent Advances in Reinforcement Learning, EWRL, pages 273–284, 2012.

[Doerr 2015] Andreas Doerr, Nathan Ratliff, Jeannette Bohg, Marc Toussaint and Stefan Schaal. *Direct Loss Minimization Inverse Optimal Control.* In Proceedings of Robotics: Science and Systems, Rome, Italy, July 2015.

[Dragan 2013] Anca D. Dragan and Siddhartha S. Srinivasa. *A policy-blending formalism for shared control*. I. J. Robotic Res., vol. 32, no. 7, pages 790–805, 2013.

[Duan 2016] Yan Duan, Xi Chen, Rein Houthooft, John Schulman and Pieter Abbeel. *Benchmarking Deep Reinforcement Learning for Continuous Control*. In Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML'16, pages 1329–1338, 2016.

[Duan 2017] Yan Duan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel and Wojciech Zaremba. *One-Shot Imitation Learning*. CoRR, vol. abs/1703.07326, 2017.

[Duda 2000] Richard O. Duda, Peter E. Hart and David G. Stork. Pattern classification (2nd edition). Wiley-Interscience, 2000.

[Elhamifar 2013] Ehsan Elhamifar and René Vidal. *Sparse Subspace Clustering: Algorithm, Theory, and Applications*. IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 11, pages 2765–2781, 2013.

[Engel 2005] Yaakov Engel, Peter Szabó and Dmitry Volkinshtein. *Learning to Control an Octopus Arm with Gaussian Process Temporal Difference Methods*. In NIPS, 2005.

[Ephraim 2002] Y Ephraim and N Merhav. *Hidden Markov processes*. IEEE Transactions on Information Theory, vol. 48, no. 6, pages 1518–1569, 2002.

[fang Wang 2016] Guo fang Wang, Zhou Fang, Ping Li and Bo Li. *Transferring knowledge from human-demonstration trajectories to reinforcement learning*. Transactions of the Institute of Measurement and Control, 2016.

[Field 2015] M. Field, D. Stirling, Z. Pan and F. Naghdy. *Learning Trajectories for Robot Programing by Demonstration Using a Coordinated Mixture of Factor Analyzers*. IEEE Trans. on Cybernetics, 2015.

[Figueroa 2017] N. Figueroa and A. Billard. *Learning Complex Manipulation Tasks from Heterogeneous and Unstructured Demonstrations*. IROS Workshop on Synergies between Learning and Interaction, 2017.

[Finn 2016] Chelsea Finn, Sergey Levine and Pieter Abbeel. *Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization*. In Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016, pages 49–58, 2016.

[Finn 2017] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel and Sergey Levine. *One-Shot Visual Imitation Learning via Meta-Learning.* CoRR, vol. abs/1709.04905, 2017.

[Fodor 2002] Imola Fodor. *A Survey of Dimension Reduction Techniques.* Technical report, 2002.

[Fox 2008] Emily B. Fox, Erik B. Sudderth, Michael I. Jordan and Alan S. Willsky. *An HDP-HMM for Systems with State Persistence.* In Proceedings of the 25th International Conference on Machine Learning, ICML '08, pages 312–319, 2008.

[Friedman 1997] Nir Friedman, Dan Geiger and Moises Goldszmidt. *Bayesian Network Classifiers.* Mach. Learn., vol. 29, no. 2-3, pages 131–163, 1997.

[Gales 1998] M.J.F. Gales. *Maximum Likelihood Linear Transformations for HMM-Based Speech Recognition.* Computer Speech and Language, vol. 12, pages 75–98, 1998.

[Gales 1999] Mark J. F. Gales. *Semi-tied covariance matrices for hidden Markov models.* IEEE Transactions on Speech and Audio Processing, vol. 7, no. 3, pages 272–281, 1999.

[Gancet 2015] J. Gancet, D. Urbina, P. Letier, M. Ilzokvitz, P. Weiss, F. Gauch, G. Antonelli, G. Indiveri, G. Casalino, A. Birk, M. F. Pfingsthorn, S. Calinon, A. Tanwani, A. Turetta, C. Walen and L. Guilpain. *DexROV: Dexterous Undersea Inspection and Maintenance in Presence of Communication Latencies.* IFAC-PapersOnLine, vol. 48, no. 2, pages 218 – 223, 2015.

[Gancet 2016] J. Gancet, P. Weiss, G. Antonelli, M. F. Pfingsthorn, S. Calinon, A. Turetta, C. Walen, D. Urbina, S. Govindaraj, P. Letier, X. Martinez, J. Salini, B. Chemisky, G. Indiveri, G. Casalino, P. Di Lillo, E. Simetti, D. De Palma, A. Birk, A. K. Tanwani, I. Havoutis, A. Caffaz and L. Guilpain. *Dexterous Undersea Interventions with Far Distance Onshore Supervision: the DexROV Project.* In IFAC Conference on Control Applications in Marine Systems (CAMS), pages 414–419, 2016.

[Garg 2016] Animesh Garg. *Optimization and Design for Automation of Brachytherapy Delivery and Learning Robot-Assisted Surgical Sub-Tasks.* PhD thesis, University of California, Berkeley, 2016.

[Ghahramani 1994] Z. Ghahramani and M. I. Jordan. *Supervised learning from incomplete data via an EM approach.* In Advances in Neural Information Processing Systems, volume 6, pages 120–127, 1994.

[Ghahramani 1997] Z. Ghahramani and G. E. Hinton. *The EM Algorithm for Mixtures of Factor Analyzers.* Technical report, University of Toronto, 1997.

[Ghahramani 2002] Zoubin Ghahramani. *Hidden Markov Models*. chapter An Introduction to Hidden Markov Models and Bayesian Networks, pages 9–42. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2002.

[Gijsberts 2013] Arjan Gijsberts and Giorgio Metta. *Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression*. Neural Networks, vol. 41, pages 59 – 69, 2013. Special Issue on Autonomous Learning.

[Goel 2001] Nagendra K. Goel and Ramesh A. Gopinath. *Multiple linear transforms*. In IEEE Intl Conf. on Acoustics, Speech, and Signal Processing, ICASSP, pages 481–484, 2001.

[Goodfellow 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio. *Generative Adversarial Nets*. In Z Ghahramani, M Welling, C Cortes, N D Lawrence and K Q Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 2672–2680. 2014.

[Gowrishankar 2013] Ganesh Gowrishankar and Etienne Burdet. *Motor planning explains human behaviour in tasks with multiple solutions*. Robotics and Autonomous Systems, vol. 61, no. 4, pages 362–368, 2013.

[Gross 2001] R. Gross and J. Shi. *The CMU Motion of Body (MoBo) Database*. Technical report CMU-RI-TR-01-18, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, June 2001.

[Guenter 2007] F. Guenter, M. Hersch, S. Calinon and A. Billard. *Reinforcement Learning for Imitating Constrained Reaching Movements*. RSJ Advanced Robotics, Special Issue on Imitative Robots, vol. 21, no. 13, pages 1521–1544, 2007.

[Hauser 2013] Kris Hauser. *Recognition, prediction, and planning for assisted teleoperation of freeform tasks*. Autonomous Robots, vol. 35, no. 4, pages 241–254, 2013.

[Havoutis 2016] Ioannis Havoutis, Ajay Kumar Tanwani and Sylvain Calinon. *Online incremental learning of manipulation tasks for semi-autonomous teleoperation*. In IROS workshop on Closed Loop Grasping and Manipulation, Oct 2016.

[Ho 2016] Jonathan Ho and Stefano Ermon. *Generative Adversarial Imitation Learning*. CoRR, vol. abs/1606.03476, 2016.

[Hogan 2012] N. Hogan and D. Sternad. *Dynamic primitives of motor behavior*. Biological Cybernetics, vol. 106, no. 11–12, pages 727–739, 2012.

[Hokayem 2006] Peter F. Hokayem and Mark W. Spong. *Bilateral Teleoperation: An Historical Survey.* Automatica, vol. 42, no. 12, pages 2035–2057, 2006.

[Howard 2010] Matthew Howard, Djordje Mitrovic and Sethu Vijayakumar. *Transferring impedance control strategies between heterogeneous systems via apprenticeship learning.* In Humanoids, pages 98–105. IEEE, 2010.

[Hoyos 2016] José Hoyos, Flavio Prieto, Guillem Alenyà and Carme Torras. *Incremental Learning of Skills in a Task-Parameterized Gaussian Mixture Model.* Journal of Intelligent and Robotic Systems, vol. 82, no. 1, pages 81–99, 2016.

[Huggins 2015] Jonathan H. Huggins, Karthik Narasimhan, Ardavan Saeedi and Vikash K. Mansinghka. *JUMP-Means: Small-Variance Asymptotics for Markov Jump Processes.* In Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, pages 693–701, 2015.

[Ijspeert 2002] Auke Jan Ijspeert, Jun Nakanishi and Stefan Schaal. *Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots.* In In IEEE International Conference on Robotics and Automation (ICRA2002, pages 1398–1403, 2002.

[Ijspeert 2013] A. Ijspeert, J. Nakanishi, P Pastor, H. Hoffmann and S. Schaal. *Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors.* Neural Computation, no. 25, pages 328–373, 2013.

[Inamura 2004] T. Inamura, I. Toshima, H. Tanie and Y. Nakamura. *Embodied Symbol Emergence Based on Mimesis Theory.* Intl Journal of Robotic Research, vol. 23, no. 4-5, pages 363–377, 2004.

[Jaquier 2017] N. Jaquier and S. Calinon. *Gaussian Mixture Regression on Symmetric Positive Definite Matrices Manifolds: Application to Wrist Motion Estimation with sEMG.* In Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), 2017.

[Jiang 2012] Ke Jiang, Brian Kulis and Michael I. Jordan. *Small-Variance Asymptotics for Exponential Family Dirichlet Process Mixture Models.* In F. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 3158–3166. Curran Associates, Inc., 2012.

[Johnson 2013] Matthew J. Johnson and Alan S. Willsky. *Bayesian Nonparametric Hidden semi-Markov Models.* J. Mach. Learn. Res., vol. 14, no. 1, pages 673–701, 2013.

[Kalakrishnan 2013] M. Kalakrishnan, P. Pastor, L. Righetti and S. Schaal. *Learning objective functions for manipulation.* In 2013 IEEE International Conference on Robotics and Automation, pages 1331–1336, May 2013.

[Kawato 1999] Mitsuo Kawato. *Internal models for motor control and trajectory planning.* Current Opinion in Neurobiology, vol. 9, no. 6, pages 718–727, 1999.

[Khansari-Zadeh 2010] S. M. Khansari-Zadeh and A. G. Billard. *BM: An iterative method to learn stable non-linear dynamical systems with Gaussian mixture models.* In Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), pages 2381–2388, Anchorage, Alaska, USA, May 2010.

[Khansari-Zadeh 2011] S. M. Khansari-Zadeh and A. Billard. *Learning Stable Non-Linear Dynamical Systems with Gaussian Mixture Models.* IEEE Trans. on Robotics, vol. 27, no. 5, pages 943–957, 2011.

[Kimura 1998] Hajime Kimura and Shigenobu Kobayashi. *An Analysis of Actor/Critic Algorithm Using Eligibility Traces: Reinforcement Learning with Imperfect Value Functions.* In Proc. of the 15th ICML, pages 278–286, 1998.

[Kober 2010] J. Kober, E. Oztop and J. Peters. *Reinforcement Learning to adjust Robot Movements to New Situations.* In Proceedings of Robotics: Science and Systems, Zaragoza, Spain, June 2010.

[Kober 2012] J. Kober, A. Wilhelm, E. Oztop and J. Peters. *Reinforcement learning to adjust parametrized motor primitives to new situations.* Autonomous Robots, April 2012.

[Kober 2013] Jens Kober, J. Andrew Bagnell and Jan Peters. *Reinforcement learning in robotics: A survey.* I. J. Robotics Res., vol. 32, no. 11, pages 1238–1274, 2013.

[Konda 2003] Vijay R. Konda and John N. Tsitsiklis. *On Actor-Critic Algorithms.* SIAM Journal on Control and Optimization, vol. 42, no. 4, pages 1143–1166, 2003.

[Kormushev 2010] P Kormushev, S Calinon and D G Caldwell. *Robot Motor Skill Coordination with EM-based Reinforcement Learning.* In Proc. {IEEE/RSJ} Intl Conf. on Intelligent Robots and Systems ({IROS}), pages 3232–3237, 2010.

[Kormushev 2013] Petar Kormushev, Sylvain Calinon and Darwin G. Caldwell. *Reinforcement Learning in Robotics: Applications and Real-World Challenges.* Robotics, vol. 2, no. 3, pages 122–148, 2013.

[Krishnamurthy 1991] Vikram Krishnamurthy, John B. Moore and Shin-Ho Chung. *On hidden fractal model signal processing.* Signal Processing, vol. 24, no. 2, pages 177–192, 1991.

[Krishnan 2017] S. Krishnan, R. Fox, I. Stoica and K. Goldberg. *DDCO: Discovery of Deep Continuous Options forRobot Learning from Demonstrations.* CoRR, 2017.

[Krishnan 2018] Sanjay Krishnan, Animesh Garg, Sachin Patil, Colin Lea, Gregory Hager, Pieter Abbeel and Ken Goldberg. Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning, pages 91–110. Springer International Publishing, Cham, 2018.

[Kronander 2011] K. Kronander, M. S. M. Khansari-Zadeh and A. Billard. *Learning to control planar hitting motions in a minigolf-like task.* In Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), pages 710–717, 2011.

[Kronander 2015] Klas Kronander, Mohammad Khansari and Aude Billard. *Incremental motion learning with locally modulated dynamical systems.* Robotics and Autonomous Systems, vol. 70, pages 52–62, 2015.

[Krueger 2010] V. Krueger, D. L. Herzog, S. Baby, A. Ude and D. Kragic. *Learning Actions from Observations: Primitive-Based Modeling and Grammar.* IEEE Robotics and Automation Magazine, vol. 17, no. 2, pages 30–43, 2010.

[Kulic 2008] D. Kulic, W. Takano and Y. Nakamura. *Incremental Learning, Clustering and Hierarchy Formation of Whole Body Motion Patterns using Adaptive Hidden Markov Chains.* Intl Journal of Robotics Research, vol. 27, no. 7, pages 761–784, 2008.

[Kulis 2012] Brian Kulis and Michael I. Jordan. *Revisiting k-means: New Algorithms via Bayesian Nonparametrics.* In Proceedings of the 29th International Conference on Machine Learning (ICML-12), pages 513–520, New York, NY, USA, 2012. ACM.

[Lee 2010a] D. Lee and C. Ott. *Incremental Motion Primitive Learning by Physical Coaching Using Impedance Control.* In Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), pages 4133–4140, Taipei, Taiwan, October 2010.

[Lee 2010b] Dongheui Lee, Christian Ott and Yoshihiko Nakamura. *Mimetic Communication Model with Compliant Physical Contact in Human - Humanoid Interaction.* I. J. Robotic Res., vol. 29, no. 13, pages 1684–1704, 2010.

[Leggetter 1995] Christopher J. Leggetter. *Improved acoustic modelling for HMMs using linear transformations.* PhD thesis, University of Cambridge, 1995.

[Levine 2012] Sergey Levine and Vladlen Koltun. *Continuous Inverse Optimal Control with Locally Optimal Examples.* In Proceedings of the 29th international conference on Machine learning, ICML, 2012.

[Levine 2015] Sergey Levine, Chelsea Finn, Trevor Darrell and Pieter Abbeel. *End-to-End Training of Deep Visuomotor Policies.* CoRR, vol. abs/1504.00702, 2015.

[Lew 2012] E. Lew, R. Chavarriaga, S. Silvoni and J. d. R. Millan. *Detection of Self-Paced Reaching Movement Intention from EEG Signals.* Frontiers in Neuroengineering, vol. 5, page 13, 2012.

[Li 2003] Ming Li and A. M. Okamura. *Recognition of operator motions for real-time assistance using virtual fixtures.* In Proc. of 11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, pages 125–131, 2003.

[Li 2004] Weiwei Li and Emanuel Todorov. *Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems.* In ICINCO, pages 222–229, 2004.

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

[Maeda 2015] G. Maeda, G. Neumann, M. Ewerton, R. Lioutikov and J. Peters. *A Probabilistic Framework for Semi-Autonomous Robots Based on Interaction Primitives with Phase Estimation.* In International Symposium of Robotics Research, 2015.

[Malla 2017] Naresh Malla and Zhen Ni. *A new history experience replay design for model-free adaptive dynamic programming.* Neurocomputing, vol. 266, pages 141 – 149, 2017.

[Manschitz 2016] S. Manschitz, M. Gienger, J. Kober and J. Peters. *Probabilistic decomposition of sequential force interaction tasks into Movement Primitives.* In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3920–3927, Oct 2016.

[Marhasev 2006] Einat Marhasev, Meirav Hadad and Gal A. Kaminka. *Non-stationary Hidden Semi Markov Models in Activity Recognition.* In Proceedings of the AAAI Workshop on Modeling Others from Observations (MOO-06), 2006.

[McLachlan 2003] G. J. McLachlan, D. Peel and R. W. Bean. *Modelling high-dimensional data by mixtures of factor analyzers.* Computational Statistics and Data Analysis, vol. 41, no. 3-4, pages 379–388, 2003.

[McNicholas 2008] P. D. McNicholas and T. B. Murphy. *Parsimonious Gaussian mixture models.* Statistics and Computing, vol. 18, no. 3, pages 285–296, September 2008.

[Medina R. 2017] Jose Medina R. and Aude Billard. *Learning Stable Task Sequences from Demonstration with Linear Parameter Varying Systems and Hidden Markov Models.* In Conference on Robot Learning (CoRL), 2017.

[Medina 2011] J. Medina, M. Lawitzky, A. Mortl, D. Lee and S. Hirche. *An Experience-Driven Robotic Assistant Acquiring Human Knowledge to Improve Haptic Cooperation.* In Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), pages 2416–2422, 2011.

[Miyamoto 1996] H. Miyamoto, F. Gandolfo, H. Gomi, S. Schaal, Y. Koike, O. Rieka, E. Nakano, Y. Wada and M. Kawato. *a kendama learning robot based on a dynamic optimiation principle.* In preceedings of the international conference on neural information processing, pages 938–942, 1996.

[Mnih 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra and Martin Riedmiller. *Playing Atari With Deep Reinforcement Learning.* In NIPS Deep Learning Workshop. 2013.

[Mnih 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg and Demis Hassabis. *Human-level control through deep reinforcement learning.* Nature, vol. 518, no. 7540, pages 529–533, 2015.

[Mombaur 2010] Katja Mombaur, Anh Truong and Jean-Paul Laumond. *From Human to Humanoid Locomotion–an Inverse Optimal Control Approach.* Autonomous Robots, vol. 28, no. 3, pages 369–383, April 2010.

[Muelling 2014] Katharina Muelling, Abdeslam Boularias, Betty Mohler, Bernhard Schölkopf and Jan Peters. *Learning strategies in table tennis using inverse reinforcement learning.* Biological Cybernetics, vol. 108, no. 5, pages 603–619, 2014.

[Mühlig 2012] M. Mühlig, M. Gienger and J. Steil. *Interactive imitation learning of object movement skills.* Autonomous Robots, vol. 32, no. 2, pages 97–114, 2012.

[Murphy 2012] Kevin P. Murphy. Machine learning: A probabilistic perspective. The MIT Press, 2012.

[Musallam 2004] S Musallam, B D Corneil, B Greger, H Scherberger and R A Andersen. *Cognitive Control Signals for Neural Prosthetics.* Science, vol. 305, no. 5681, pages 258–262, 2004.

[Mussa-Ivaldi 1994] F. A. Mussa-Ivaldi, S. F. Giszter and E. Bizzi. *Linear combinations of primitives in vertebrate motor control.* Proc. National Academy of Sciences, vol. 91, pages 7534–7538, 1994.

[Neal 1999] R. M. Neal and G. E. Hinton. *A view of the EM algorithm that justifies incremental, sparse, and other variants.* In Learning in graphical models, pages 355–368. MIT Press, Cambridge, MA, USA, 1999.

[Nehaniv 2004] Chrystopher L. Nehaniv and Kerstin Dautenhahn, editors. Imitation and social learning in robots, humans, and animals: behavioural, social and communicative dimensions. Cambridge University Press, 2004.

[Neu 2012] Gergely Neu and Csaba Szepesvári. *Apprenticeship Learning using Inverse Reinforcement Learning and Gradient Methods.* CoRR, vol. abs/1206.5264, 2012.

[Ng 2000] Andrew Y. Ng and Stuart Russell. *Algorithms for Inverse Reinforcement Learning.* In Proceedings of the 17th International Conference on Machine Learning, ICML, pages 663–670, 2000.

[Nguyen-Tuong 2009] D. Nguyen-Tuong, M. Seeger and J. Peters. *Model Learning with Local Gaussian Process Regression.* Advanced Robotics, vol. 23, no. 15, pages 2015–2034, 2009.

[Nguyen-Tuong 2011] D. Nguyen-Tuong and J. Peters. *Model learning for robot control: a survey.* Cognitive Processing, vol. 12, no. 4, pages 319–340, 2011.

[Niazi 2011] Imran Khan Niazi, Ning Jiang, Olivier Tiberghien, Jørgen Feldbæk Nielsen, Kim Dremstrup and Dario Farina. *Detection of movement intention from single-trial movement-related cortical potentials.* Journal of Neural Engineering, vol. 8, no. 6, page 66009, 2011.

[Niekum 2012] Scott Niekum, Sarah Osentoski, George Konidaris and Andrew G Barto. *Learning and Generalization of Complex Tasks from Unstructured Demonstrations.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 5239–5246, 2012.

[Niemeyer 2008] Günter Niemeyer, Carsten Preusche and Gerd Hirzinger. Telerobotics, pages 741–757. Springer Berlin Heidelberg, 2008.

[Nolin 2003] J. T. Nolin, P. M. Stemniski and A. M. Okamura. *Activation cues and force scaling methods for virtual fixtures.* In Proc. of 11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, pages 404–409, 2003.

[Olsen 2004] Peder A. Olsen and Ramesh A. Gopinath. *Modeling inverse covariance matrices by basis expansion.* IEEE Transactions on Speech and Audio Processing, vol. 12, no. 1, pages 37–46, 2004.

[Paraschos 2013] Alexandros Paraschos, Christian Daniel, Jan R Peters and Gerhard Neumann. *Probabilistic Movement Primitives.* In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 26, pages 2616–2624. Curran Associates, Inc., 2013.

[Pastor 2012] P. Pastor, M. Kalakrishnan, L. Righetti and S. Schaal. *Towards Associative Skill Memories.* In 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012), pages 309–315, Nov 2012.

[Pathak 2010] Kaustubh Pathak, Andreas Birk, Narunas Vaskevicius and Jann Poppinga. *Fast Registration Based on Noisy Planes With Unknown Correspondences for 3-D Mapping.* IEEE Trans. Robotics, vol. 26, no. 3, pages 424–441, 2010.

[Pentland 1999] Alex Pentland and Andrew Liu. *Modeling and Prediction of Human Behavior.* Neural Comput., vol. 11, no. 1, pages 229–242, January 1999.

[Peters 2007] Jan Peters and Stefan Schaal. *Reinforcement learning by reward-weighted regression for operational space control.* In Proceedings of the 24th international conference on Machine learning, ICML '07, pages 745–750, 2007.

[Peters 2008] Jan Peters and Stefan Schaal. *Reinforcement learning of motor skills with policy gradients.* Neural Networks, vol. 21, no. 4, pages 682–697, 2008.

[Pfingsthorn 2016] Max Pfingsthorn, Ravi Rathnam, Tomasz Luczynski and Andreas Birk. *Full 3D Navigation Correction using Low Frequency Visual Tracking with a Stereo Camera.* In IEEE Oceans, 2016.

[Pignat 2017] E. Pignat and S. Calinon. *Learning adaptive dressing assistance from human demonstration.* Robotics and Autonomous Systems, vol. 93, pages 61–75, July 2017.

[Pitman 2002] Jim Pitman. *Poisson-Dirichlet and GEM Invariant Distributions for Split-and-Merge Transformations of an Interval Partition.* Combinatorics, Probability and Computing, vol. 11, pages 501–514, 2002.

[R. Li 2010] S. Sclaroff R. Li T. Tian and M. Yang. *3D Human Motion Tracking with a Coordinated Mixture of Factor Analyzers.* International Journal of Computer Vision, vol. 87, no. 1-2, pages 170–190, 2010.

[Rabiner 1989] L. R. Rabiner. *A tutorial on hidden Markov models and selected applications in speech recognition.* Proc. IEEE, vol. 77:2, pages 257–285, 1989.

[Raiola 2015] Gennaro Raiola, Xavier Lamy and Freek Stulp. *Co-manipulation with multiple probabilistic virtual guides.* In Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), pages 7–13, 2015.

[Ramachandran 2007] Deepak Ramachandran and Eyal Amir. *Bayesian inverse reinforcement learning*. In Proceedings of the 20th international joint conference on Artifical intelligence, IJCAI, pages 2586–2591, 2007.

[Ramesh 1992] P. Ramesh and J. G. Wilpon. *Modeling state durations in hidden Markov models for automatic speech recognition*. In IEEE International Conference on Acoustics, Speech, and Signal Processing, volume 1, pages 381–384, 1992.

[Rasmussen 2006] Carl E. Rasmussen and Christopher Williams. Gaussian Processes for Machine Learning. MIT Press, 2006.

[Ratliff 2006] Nathan D. Ratliff, J. Andrew Bagnell and Martin A. Zinkevich. *Maximum margin planning*. In Proceedings of the 23rd International Conference on Machine Learning, ICML, 2006.

[Ratliff 2009] Nathan D. Ratliff, David Silver and J. Andrew Bagnell. *Learning to search: Functional gradient techniques for imitation learning*. Autonomous Robots, vol. 27, no. 1, pages 25–53, Jul 2009.

[Rosenberg 1993] Louis B. Rosenberg. *Virtual Fixtures: Perceptual Tools for Telerobotic Manipulation*. In IEEE Virtual Reality Annual International Symposium, VRAIS, pages 76–82, 1993.

[Roweis 1999] Sam Roweis and Zoubin Ghahramani. *A Unifying Review of Linear Gaussian Models*. Neural Comput., vol. 11, no. 2, pages 305–345, February 1999.

[Roychowdhury 2013] Anirban Roychowdhury, Ke Jiang and Brian Kulis. *Small-Variance Asymptotics for Hidden Markov Models*. In Advances in Neural Information Processing Systems 26, pages 2103–2111. Curran Associates, Inc., 2013.

[Rozo 2016] L. Rozo, S. Calinon, D. G. Caldwell, P. Jimenez and C. Torras. *Learning Physical Collaborative Robot Behaviors from Human Demonstrations*. IEEE Trans. on Robotics, vol. 32, no. 3, pages 513–527, 2016.

[Rusu 2016] Andrei A. Rusu, Matej Vecerik, Thomas Rothörl, Nicolas Heess, Razvan Pascanu and Raia Hadsell. *Sim-to-Real Robot Learning from Pixels with Progressive Nets*. CoRR, vol. abs/1610.04286, 2016.

[Salakhutdinov 2007] Ruslan Salakhutdinov, Andriy Mnih and Geoffrey Hinton. *Restricted Boltzmann Machines for Collaborative Filtering*. In Proceedings of the 24th International Conference on Machine Learning, ICML '07, pages 791–798. ACM, 2007.

[Schaal 2003] S. Schaal, A. Ijspeert and A. Billard. *Computational approaches to motor learning by imitation.* Philosophical Transaction of the Royal Society of London: Series B, Biological Sciences, vol. 358, no. 1431, pages 537–547, 2003.

[Sermanet 2016] Pierre Sermanet, Kelvin Xu and Sergey Levine. *Unsupervised Perceptual Rewards for Imitation Learning.* CoRR, vol. abs/1612.06699, 2016.

[Sermanet 2017] Pierre Sermanet, Corey Lynch, Jasmine Hsu and Sergey Levine. *Time-Contrastive Networks: Self-Supervised Learning from Multi-View Observation.* CoRR, vol. abs/1704.06888, 2017.

[Sethuraman 1994] Jayaram Sethuraman. *A constructive definition of Dirichlet priors.* Statistica Sinica, vol. 4, pages 639–650, 1994.

[Sheridan 1992] Thomas B. Sheridan. Telerobotics, automation, and human supervisory control. MIT Press, Cambridge, MA, USA, 1992.

[Sheridan 1995] T.B. Sheridan. *Teleoperation, telerobotics and telepresence: A progress report.* Control Engineering Practice, vol. 3, no. 2, pages 205 – 214, 1995.

[Silver 2016] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel and Demis Hassabis. *Mastering the Game of Go with Deep Neural Networks and Tree Search.* Nature, vol. 529, no. 7587, pages 484–489, 2016.

[Silverio 2015] J. Silverio, L. Rozo, S. Calinon and D. G. Caldwell. *Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems.* In Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), pages 464–470, 2015.

[Simetti 2017] E. Simetti, F. Wanderlingh, G. Casalino, G. Indiveri and G. Antonelli. *DexROV project: Control framework for underwater interaction tasks.* In OCEANS 2017 - Aberdeen, pages 1–6, June 2017.

[Smola 2004] Alex J. Smola and Bernhard Schölkopf. *A Tutorial on Support Vector Regression.* Statistics and Computing, vol. 14, no. 3, pages 199–222, 2004.

[Song 2005] M. Song and H. Wang. *Highly efficient incremental estimation of Gaussian mixture models for online data stream clustering.* In Proc. of SPIE: Intelligent Computing - Theory and Applications III, volume 5803, pages 174–183, 2005.

[Spriggs 2009] E. H. Spriggs, F. De La Torre and M. Hebert. *Temporal segmentation and activity classification from first-person sensing.* In IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pages 17–24, 2009.

[Stolle 2002] Martin Stolle and Doina Precup. Learning options in reinforcement learning, pages 212–223. Springer Berlin Heidelberg, 2002.

[Stulp 2015] Freek Stulp and Olivier Sigaud. *Many regression algorithms, one unified model - A review.* Neural Networks, page 28, 2015.

[Sugimoto 2013] N. Sugimoto and J. Morimoto. *Trajectory-model-based reinforcement learning: Application to bimanual humanoid motor learning with a closed-chain constraint.* In 2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids), pages 429–434, Oct 2013.

[Sugiura 2011] K. Sugiura, N. Iwahashi, H. Kashioka and S. Nakamura. *Learning, Generation, and Recognition of Motions by Reference-Point-Dependent Probabilistic Models.* Advanced Robotics, vol. 25, no. 5, 2011.

[Sutton 1992] Richard S. Sutton, Andrew G. Barto and Ronald J. Williams. *Reinforcement Learning is Direct Adaptive Optimal Control.* In In Proceedings of the American Control Conference, pages 2143–2146, 1992.

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

[Syed 2008] Umar Syed, Robert Schapire and Michael Bowling. *Apprenticeship Learning Using Linear Programming.* In Proceedings of the Twenty-Fifth International Conference on Machine Learning, pages 1032–1039, 2008.

[Szepesvári 2010] Csaba Szepesvári. Algorithms for reinforcement learning. Morgan & Claypool, 2010.

[Tang 2012] Y. Tang, R. Salakhutdinov and G. Hinton. *Deep Mixtures of Factor Analysers.* In Proc. Intl Conf. on Machine Learning (ICML), Edinburgh, Scotland, 2012.

[Tanwani 2011] A. K. Tanwani. Optimizing walking of a humanoid robot using reinforcement learning. MS thesis, Warsaw University of Technology, 2011.

[Tanwani 2013a] A. K. Tanwani and A. Billard. *Inverse Reinforcement Learning for Compliant Manipulation in Letter Handwriting.* In National Centre of Competence in Research in Robotics, 2013.

[Tanwani 2013b] A. K. Tanwani and A. Billard. *Transfer in inverse reinforcement learning for multiple strategies.* In Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), pages 3244–3250, 2013.

[Tanwani 2014] A. K. Tanwani, J. d. R. Millán and A Billard. *Rewards-driven control of robot arm by decoding EEG signals.* In Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE, pages 1658–1661, 2014.

[Tanwani 2016a] A. K. Tanwani and S. Calinon. *Learning Robot Manipulation Tasks With Task-Parameterized Semitied Hidden Semi-Markov Model.* Robotics and Automation Letters, IEEE, vol. 1, no. 1, pages 235–242, 2016.

[Tanwani 2016b] A. K. Tanwani and S. Calinon. *Online Inference in Bayesian Non-Parametric Mixture Models under Small Variance Asymptotics.* In NIPS workshop on Advances in Approximate Bayesian Inference, pages 1–5, 2016.

[Tanwani 2016c] A. K. Tanwani and S. Calinon. *Small Variance Asymptotics for Non-Parametric Online Robot Learning.* CoRR, vol. abs/1610.0, 2016. Int. J. of Robotics and Research (condtionally accepted).

[Tanwani 2017] A. K. Tanwani and S. Calinon. *A Generative Model for Intention Recognition and Manipulation Assistance in Teleoperation.* In Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), pages 1–8, 2017.

[Teh 2006] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal and David M. Blei. *Hierarchical Dirichlet processes.* Journal of the American Statistical Association, vol. 101, no. 476, pages 1566–1581, 2006.

[Theodorou 2010] Evangelos Theodorou, Jonas Buchli and Stefan Schaal. *A Generalized Path Integral Control Approach to Reinforcement Learning.* J. Mach. Learn. Res., vol. 11, pages 3137–3181, 2010.

[Tipping 1999] M. E. Tipping and C. M. Bishop. *Mixtures of probabilistic principal component analyzers.* Neural Computation, vol. 11, no. 2, pages 443–482, 1999.

[Todorov 2002] E. Todorov and M. I. Jordan. *A Minimal Intervention Principle for Coordinated Movement.* In Advances in Neural Information Processing Systems (NIPS), pages 27–34, 2002.

[Ude 2010] A. Ude, A. Gams, T. Asfour and J. Morimoto. *Task-Specific Generalization of Discrete and Periodic Dynamic Movement Primitives.* IEEE Trans. on Robotics, vol. 26, no. 5, pages 800–815, 2010.

[Ureche 2015] A.L.P. Ureche, K. Umezawa, Y. Nakamura and A. Billard. *Task Parameterization Using Continuous Constraints Extracted From Human Demonstrations.* Robotics, IEEE Transactions on, vol. 31, no. 6, pages 1458–1471, 2015.

[Vakanski 2012] A. Vakanski, I. Mantegh, A. Irish and F. Janabi-Sharifi. *Trajectory Learning for Robot Programming by Demonstration Using Hidden Markov Model and Dynamic Time Warping.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 42, no. 4, pages 1039–1052, 2012.

[Van Gael 2008] Jurgen Van Gael, Yunus Saatci, Yee Whye Teh and Zoubin Ghahramani. *Beam Sampling for the Infinite Hidden Markov Model.* In Proceedings of the 25th International Conference on Machine Learning, ICML '08, pages 1088–1095, New York, NY, USA, 2008.

[Vijayakumar 2000] Sethu Vijayakumar and Stefan Schaal. *Locally Weighted Projection Regression: An $O(n)$ Algorithm for Incremental Real Time Learning in High Dimensional Space.* In Proc. 17th International Conf. on Machine Learning, pages 1079–1086. Morgan Kaufmann, San Francisco, CA, 2000.

[Vijayakumar 2005] S. Vijayakumar, A. D'souza and S. Schaal. *Incremental Online Learning in High Dimensions.* Neural Computation, vol. 17, no. 12, pages 2602–2634, 2005.

[Vogel 2016] J. Vogel, K. Hertkorn, R. U. Menon and M. A. Roa. *Flexible, semi-autonomous grasping for assistive robotics.* In Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), pages 4872–4879, 2016.

[Waldert 2008] Stephan Waldert, Hubert Preissl, Evariste Demandt, Christoph Braun, Niels Birbaumer, Ad Aertsen and Carsten Mehring. *Hand Movement Direction Decoded from MEG and EEG.* The Journal of Neuroscience, vol. 28, no. 4, pages 1000–1008, 2008.

[Wan 2000] E.A. Wan and R. Van Der Merwe. *The Unscented Kalman Filter for Nonlinear Estimation.* pages 153–158, 2000.

[Wang 2013] Z. Wang, K. Mülling, M. Deisenroth, H. Ben Amor, D. Vogt, B. Schölkopf and J. Peters. *Probabilistic movement modeling for intention inference in human-robot interaction.* International Journal of Robotics Research, vol. 32, no. 7, pages 841–858, 2013.

[Wang 2015] Yining Wang and Jun Zhu. *DP-space: Bayesian Nonparametric Subspace Clustering with Small-variance Asymptotics.* In Proc. of the 32nd International Conference on Machine Learning, ICML, pages 862–870, 2015.

[Wawrzyński 2009] P. Wawrzyński. *Real-time reinforcement learning by sequential actor-critics and experience replay.* Neural Networks, vol. 22, pages 1484–1497, 2009.

[Wawrzynski 2013] Pawel Wawrzynski and Ajay Kumar Tanwani. *Autonomous reinforcement learning with experience replay.* Neural Networks, vol. 41, no. 0, pages 156–167, 2013.

[Weber 2000] M. Weber, M. Welling and P. Perona. Unsupervised learning of models for recognition, pages 18–32. Springer Berlin Heidelberg, 2000.

[Wilson 1999] A. D. Wilson and A. F. Bobick. *Parametric Hidden Markov Models for Gesture Recognition.* IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 21, no. 9, pages 884–900, 1999.

[Wolpert 2011] D. M. Wolpert, J. Diedrichsen and J. R. Flanagan. *Principles of sensorimotor learning.* Nature Reviews, vol. 12, pages 739–751, 2011.

[Yamazaki 2005] T. Yamazaki, N. Niwase, J. Yamagishi and T. Kobayashi. *Human walking motion synthesis based on multiple regression hidden semi-Markov model.* In Proc. Intl Conf. on Cyberworlds, pages 445–452, 2005.

[Yang 2015] Y. Yang, V. Ivan and S. Vijayakumar. *Real-time motion adaptation using relative distance space representation.* In International Conference on Advanced Robotics (ICAR), pages 21–27, 2015.

[Yekutieli 2005] Yoram Yekutieli, Roni Sagiv-Zohar, Ranit Aharonov, Yaakov Engel, Binyamin Hochner and Tamar Flash. *Dynamic model of the octopus arm. I. Biomechanics of the octopus reaching movement.* J Neurophysiol, vol. 94, no. 2, pages 1443–58, 2005.

[Yeung 2016] Serena Yeung, Olga Russakovsky, Greg Mori and Li Fei-Fei. *End-to-End Learning of Action Detection from Frame Glimpses in Videos.* In IEEE Conference on Computer Vision and Pattern Recognition CVPR, pages 2678–2687, 2016.

[Yin 2016] Hang Yin, Patrícia Alves-Oliveira, Francisco S. Melo, Aude Billard and Ana Paiva. *Synthesizing Robotic Handwriting Motion by Learning from Human Demonstrations.* In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI'16, pages 3530–3537. AAAI Press, 2016.

[Yoerger 1987] D. Yoerger and J. J. Slotine. *Supervisory control architecture for underwater teleoperation.* In IEEE International Conference on Robotics and Automation., volume 4, pages 2068–2073, Mar 1987.

[Yu 2005] Wentao Yu, R. Alqasemi, R. Dubey and N. Pernalete. *Telemanipulation Assistance Based on Motion Intention Recognition.* In IEEE International Conference on Robotics and Automation, pages 1121–1126, 2005.

[Yu 2006] S.-Z. Yu and H. Kobayashi. *Practical Implementation of an Efficient Forward-Backward Algorithm for an Explicit-Duration Hidden Markov Model.* IEEE Trans. on Signal Processing, vol. 54, no. 5, pages 1947–1951, 2006.

[Yu 2010] S.-Z. Yu. *Hidden semi-Markov models.* Artificial Intelligence, vol. 174, pages 215–243, 2010.

[Zeestraten 2016] M. Zeestraten, S. Calinon and D. G. Caldwell. *Variable Duration Movement Encoding with Minimal Intervention Control.* In ICRA, pages 497–503, Stockholm, Sweden, May 2016.

[Zen 2007] H. Zen, K. Tokuda and T. Kitamura. *Reformulating the HMM as a trajectory model by imposing explicit relationships between static and dynamic feature vector sequences.* Computer Speech and Language, vol. 21, no. 1, pages 153–173, 2007.

[Zhang 2004] Zhihua Zhang, Kap Luk Chan, James T. Kwok and Dit-Yan Yeung. *Bayesian Inference on Principal Component Analysis Using Reversible Jump Markov Chain Monte Carlo.* In Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA, pages 372–377, 2004.

[Zhang 2017] T. Zhang, Z. McCarthy, O. Jow, D. Lee, K. Goldberg and P. Abbeel. *Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation.* ArXiv e-prints, 2017.

[Zhu 2009] Xiaojin Zhu, Andrew B. Goldberg, Ronald Brachman and Thomas Dietterich. Introduction to semi-supervised learning. Morgan and Claypool Publishers, 2009.

[Ziebart 2008] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell and Anind K. Dey. *Maximum entropy inverse reinforcement learning.* In Proceedings of the 23rd national conference on Artificial intelligence - Volume 3, AAAI'08, pages 1433–1438, 2008.

Ajay Kumar Tanwani

http://www.ajaytanwani.com
aktanwani@gmail.com

Rue des Echelettes 11,
1004 Lausanne, Switzerland.
Mobile: +41 789 326 336

Summary

- Passionate about developing autonomous learning systems from raw sensory data
- Interdisciplinary research experience in designing robot learning algorithms with generative models
- Interested in research and development of machine learning techniques for solving real world problems

Education

- **Ph.D. – Computer Science, École Polytechnique Fédérale de Lausanne EPFL** Lausanne, Switzerland
  Thesis: Generative models for learning robot manipulation skills from humans Sep. 2012 – Jan. 2018
- **MS – European Masters on Advanced Robotics EMARO**
  Warsaw University of Technology WUT, Poland; CGPA: 6.0 - Excellent Sep. 2010 – Nov. 2011
  University of Genova UNIGE, Italy; Marks: 106/110 Sep. 2009 – June 2010
- **BE – National University of Sciences and Technology NUST** Islamabad, Pakistan
  Mechatronics Engineering; CGPA: 3.95/4.00 Sep. 2004 – May. 2008

Work Experience

- **Google X** Mountain View, USA
  Research Intern June 2017 - Oct. 2017
  - **Problem**: robot learning from videos in the virtual reality environment; **Solution**: I developed an invariant end-to-end deep segmentation method for extracting sub-goals from unstructured demonstrations that are sequenced together with a set of high-level instructions for performing complex manipulation tasks.
- **Idiap Research Institute, École Polytechnique Fédérale de Lausanne EPFL** Lausanne, Switzerland
  Ph.D. Research May 2012 - Jan. 2018
  - **Problem**: unsupervised robot learning from human demonstrations; **Solution**: I developed algorithms for invariant task representations with hidden semi-Markov models for task segmentation and reproduction; learning in latent space to scale the mixture models; and learning online with Bayesian non-parametric mixture models under small variance asymptotics.
  - **Problem**: teleoperate robots over satellite communication; **Solution**: I developed a cognitive engine based on invariant task-parameterized generative mixture models for recognizing the intention of the user and providing assistance to remotely perform manipulation tasks by semi-autonomous teleoperation.
  - **Problem**: learn reward function(s) from multiple experts' demonstrations; **Solution**: I enclosed all experts' demonstrations driven by different reward functions in a convex set of optimal deterministic policies via inverse reinforcement learning while using transfer of knowledge to bootstrap the learning process.

- **Biometrics and Machine Learning Team, Warsaw University of Technology** Warsaw, Poland
  MS Research Jan. 2011 – Oct. 2012
  - **Problem**: learn optimal control policies in continuous space; **Solution**: I investigated actor-critics with experience replay for optimizing walking of a Bioloid humanoid robot and running of half-cheetah.
- **Next Generation Intelligent Networks Research Center, FAST-NUCES** Islamabad, Pakistan
  Research Engineer May 2008 – Sep. 2009
  - **Problem**: remote patient monitoring system with a focus on antenatal care; **Solution**: I led the project and developed an intelligent clinical decision support system by quantifying the complexity of a dataset, and choosing the appropriate data pre-processing method, classifier, classifier ensemble, and/or team of classifiers.

Skills

- **Programming**: Python; Matlab; C++, C; TensorFlow, Keras; ROS; Git; CMake; Google python style programming; unit testing
- **Technical**:
  probabilistic learning models: data pre-processing, feature selection and (subspace) clustering/classification/regression/time-series analysis with mixture models (GMMs, HMMs, HSMMs, MPPCA, MFA, semi-tied models, Bayesian non-parametric models under small variance asymptotics)
  (inverse) reinforcement learning/optimal control: learning cost functions and optimal policies (LQR/iLQR, actor-critics, maximum-entropy models, neural networks, shooting/collocation methods, dynamic programming)
  deep learning: augmenting deep learning architectures (inception) for object recognition and video segmentation
  robots: Baxter, Barrett WAM, Kuka LWR, Bioloid
- **Academic**: **teaching assistant:** Data analysis and model classification'13, Robotics Practical'14
  EU projects: DexROV; NCCR Robotics

Honors and Awards

- **Best Robotic Manipulation Paper Award Finalist** at IEEE International Conference on Robotics and Automation ICRA, 2016 [Acceptance rate: 34.7 % of 2357 submissions]
- **Winner of World Summit Youth Award** selected out of 612 submissions from 102 different countries, Mexico, 2009
- European Union Erasmus Mundus scholarship for EMARO 2009 – 2011
- Winner of Best Engineering Project in: 4th International Software Exhibition and Competition, **SOFTEC**, 2008; 7th National Computer Project Exhibition and Competition, **COMPPEC**, 2008

- Rector's Gold Medal awarded by the Prime Minister of Pakistan for Best Final Project in Mechatronics, NUST 2008
- Winner of several all Pakistan robotics competitions: National Engineering Robotics Contest **NERC**, 2007; **NASCON** Robotics Competition, FAST-NUCES 2008; First Institute of Space and Technology **IST** Robotics Contest, 2005
- Dean's Plaque of Excellence for Distinguished Student in 5th, 6th and 7th semester at NUST; Merit scholarship in all semesters at NUST

Selected Publications

- A. K. Tanwani, S. Calinon, **Small variance asymptotics for non-parametric online robot learning**, arXiv:1610.92468, Int. J. of Robotics and Research, IJRR 2017 *(conditionally accepted)*
- A. K. Tanwani, S. Calinon, **Online Inference in Bayesian non-parametric mixture models under small variance asymptotics**, Advances in Approximate Bayesian Inference, NIPS, 2016
- A. K. Tanwani, S. Calinon, **Learning robot manipulation tasks with task-parameterized semi-tied hidden semi-Markov model**, IEEE Robotics and Automation Letters, RA-L 2016 [Best Manipulation Paper Award Finalist at ICRA, 2016]
- A. K. Tanwani, J. d. R. Millan, A. Billard, **Rewards-driven control of robot arm by decoding EEG signals**, IEEE Eng. in Medicine and Biology Society Conf., EMBC 2014
- A. K. Tanwani, A. Billard, **Transfer in inverse reinforcement learning for multiple strategies**, IEEE/RSJ Conf. on Intelligent Robots and Systems, IROS 2013
- P. Wawrzynski, A. K. Tanwani, **Autonomous reinforcement learning with experience replay**, J. of Neural Networks, 2013
- A. K. Tanwani, J. Afridi, Z. Shafique, M. Farooq, **Guidelines to select machine learning scheme for classification of biomedical datasets**, 7th Evolutionary Computation, Machine Learning and Datamining Conf., EvoBIO 2009

Professional Activities

- Program committee member: **AABI (NIPS)**, 2017; Reviewer: IJRR, ICRA, IROS, Auto. Robots
- Professional Development Board Member in Erasmus+ Student and Alumni Association, **ESAA**
- Elected Program Representative of EMARO from France, Italy, and Poland, 2010 – 2011
- General Secretary of Web and Graphics Club in NUST, 2007 – 2008

Languages

- English (advanced), Urdu (advanced), French (intermediate), Italian (basic), Polish (basic)