UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Robot Learning from Demonstration with Gaussian Processes

Miguel Arduengo García

Advisors: Adrià Colomé Figueras
Carme Torras Genís

Tutor: Carlos Ocampo Martínez

A thesis submitted in partial fulfillment of the requirements for the:

Master's Degree in Industrial Engineering

Master's Degree in Automatic Control and Robotics

ETSEIB

Escola Tècnica Superior d'Enginyeria Industrial de Barcelona

July, 2021

Abstract

Autonomous systems are no longer confined to factories, but they are progressively spreading to urban, social, and assistive domains. However, in order to become handy co-workers and helpful assistants, robots must be endowed with quite different abilities than their industrial ancestors, and a lot of additional research is still required. A key challenge in intelligent robotics is creating autonomous agents that are capable of directly interacting with the world around them to achieve their goals. Learning plays a central role in intelligent autonomous systems, as the real world contains too much uncertainty and a robot must be capable of dealing with environments that neither it nor its designers have foreseen.

Learning from demonstration is a promising paradigm that allows robots to learn complex tasks that cannot be easily scripted, but can be demonstrated by a human teacher. In this thesis, we develop complete learning from demonstration framework. We first present a whole-body teleoperation approach for human motion transfer, which allows a teacher equipped with a motion capture system to intuitively provide demonstrations to a robot. Then, to learn a generalized representation of the task which can be adapted to unforeseen scenarios, we unify in a single, entirely Gaussian-Process-based formulation, the main components of a state-of-the-art method. We evaluate our approach through a series of real-world experiments with the manipulator robot TIAGo, achieving satisfactory results.

Finally, we must be aware that we are in a technological inflection point in which robots are developing the capacity to greatly increase their cognitive and physical capabilities. This will raise complex issues regarding the economy, ethics, law, and the environment, which we provide an overview of in this thesis. Intelligent robotics offer an unimaginable spectrum of possibilities, with the appropriate attention and the right policies they open the doors to new sources of value and growth. However, it is in the hands of scientists and engineers to not look away and anticipate the potential impacts in order to turn robots into the motor of global prosperity.

Contents

List of Figures

Chapter 1

Introduction

Robotics is concerned with the study of those machines that can help and assist human beings in the execution of a task, as regards both physical activity and decision making. Although one might think of robots as inventions of the modern era, over the course of the centuries it has been a constant ambition for scientists and engineers to seek artifacts able to mimic our behavior.

Since Issac Asimov started to write short stories in the 1940s, people usually imagine robots as companions with a human form who can think, see, walk, and hear. However, the first practical robots that appeared in the 1970s fell quite far from this conception. These were mechanical arms equipped with a gripper and fixed in place that manufactured products on assembly lines. Industrial robots excel at simple and repetitive tasks where speed, precision, and reliability are paramount but strive for autonomy in complex, unstructured, and unpredictable environments.

Nowadays, we have access to a much more rich set of sophisticated sensors, powerful computing platforms, and all various agile mechatronic devices. Autonomous systems are no longer confined to factories, but they are progressively spreading to urban, social, and assistive domains (Figure 1.1). Robots such as vacuum cleaners, lawnmowers, and window cleaners, as well as a huge number of toys, have been made commercially available. Nevertheless, a lot of additional research is still required. In order to become handy co-workers and helpful assistants, robots must be endowed with quite different abilities than their industrial ancestors.

In this introduction, we first provide an overview in Section 1.1 of the main challenges and research questions addressed throughout the thesis. Specifically, we focus on the relevance and need of developing methods that allow robots to easily learn new skills, for endowing them with human-like abilities. Afterward, in Section 1.2, we present the main contributions of this thesis. Finally, in Section 1.3, we outline its organization and, in Section 1.4, we list its associated publications.

Figure 1.1: From left to right: past, present, and future of robotics.

1.1 Motivation

A key challenge in intelligent robotics is creating autonomous agents that are capable of directly interacting with the world around them to achieve their goals. Learning plays a central role in such autonomous systems, as the real world contains too much uncertainty. A robot must be capable of dealing with environments that neither it nor its designers have foreseen or encountered before.

For endowing robots with close-to-human abilities, inspiration has been taken from the ways humans learn. When we learn to perform a task, we naturally shape our behavior based on our a-priori knowledge and experience. This training is commonly acquired by mimicking people that we may have seen executing similar tasks. However, we do not reproduce exactly the same gestures that we have observed. Instead, we naturally adapt our motion to make it compatible with both, the available information about the task, and our instantaneous perception of the environment.

One of the main research directions for building the future intelligent robots is founded on the learning from demonstration (LfD) paradigm. This approach is aimed at giving robots the capability of learning skills by observing and generalizing from human demonstrations. LfD is also motivated by the challenge of enabling a novice user, a non-expert programmer, to customize existing robot behaviors or develop new ones through intuitive teaching methods.

The field of learning from demonstration has already generated innumerable insights into the science and art of teaching robots to perform a variety of tasks across multiple domains. However, several challenges remain to be addressed if we aspire to allow robots to learn from humans and operate in challenging environments fluently and efficiently (Figure 1.2).

Figure 1.2: Robots performing tasks with human-like abilities (after Josh Ellingson).

Often, robots cannot act the same way as a human does, due to the differences in physical embodiment. Additionally, robots perceive the world in a fundamentally different way. While a person can feel force, rugosity, or temperature through tact, a robot might simply perceive contact. Providing a set of demonstrations that contain the necessary information requires appropriate interfaces and a deep understanding of the robot's behavior and its limitations. This first challenge that appears in LfD can be synthesized in the following question

How can a robot imitate the human performance of a task?

The next problem, once we have the demonstration data in a form that can be interpreted by the robot, is to actually learn the task. The human teacher will only show a limited number of examples. However, in order to operate autonomously in the real world, the robot must be able to deal with unforeseen scenarios that need the adaptation of the learned motions. This problem requires the development of expressive and versatile data-driven models that can exploit the information encoded in the demonstrations efficiently, leading to the following interrogation

How can a robot learn and generalize a skill from a few demonstrations?

Last but not least, we must be aware that we are in a technological inflection point in which robots are developing the capacity to greatly increase their cognitive and physical capabilities. This will raise complex issues regarding the economy, ethics, law, and the environment. It is the responsibility of scientists and engineers to ask ourselves the following question in order to turn robots into motors of prosperity

What is the potential impact of the evolution of robots in our societies?

1.2 Contributions

This thesis focuses on addressing the research questions formulated in Section 1.1, which are concerned with the learning from demonstration problem and the prospective repercussions that will come along with the expansion of automation. Specifically, we develop a complete LfD framework that encompasses from intuitively transferring human motion to a robot, to the method for learning the task and adapt to unforeseen scenarios; and review and discuss the existing literature on the potential impact of robotics. The main contributions can be listed as follows:

- **Intuitive compendium of the theoretical fundamentals:** We have put together the main concepts of machine learning, namely related to nonlinear regression and probability theory, and Gaussian Process (GP) regression. This constitutes the basis for the comprehension of the standard LfD methods and our approach. The value of this compendium lies in the mindset with which it has been written, minimizing the 'science per cognitive load' rather than the page count, presenting the theory as intuitively and illustrated as possible.
- **Comprehensive survey of trajectory-based LfD methods:** We describe a representative subset of the research that has so far been carried out on robot learning from demonstration. In particular, of the methods that use trajectory-based representations for encoding the learned skill. This survey is aimed at providing a general overview of the state-of-the-art within the field, with a special focus on understanding the main ideas behind each method.
- **Interface for transferring human motion to a mobile robot:** We present a robot whole-body teleoperation framework for human motion transfer. Our approach allows a human demonstrator, equipped with a motion capture system, to intuitively provide demonstrations to a robot. The most important aspects of our system are:
  - *General solution of the correspondence problem:* Defining a general mapping between the human posture and the robot, we overcome the problems in LfD derived from the differences in physical embodiment.
  - *Efficient control of the robot's degrees-of-freedom:* We achieve tight coordination between the motion of the robot links and an effective real-time imitation by adopting a whole-body control scheme.
  - *Safe physical human-robot interaction:* We design a variable admittance controller for stably adapting the end-effector dynamics to switch between stiff and compliant behaviors. This allows a tight position control while ensuring safety when physically interacting with a human.

- **Novel Gaussian-Process-based LfD method:** We unify in a single, entirely Gaussian-Process-based framework, the main components required for state-of-the-art learning from demonstration method. This expressive and versatile data-driven representation allows to generalize over multiple demonstrations, and also encode the uncertainty along the different stages of the task. The most relevant features of our approach are:
  - *High-dimensional learning:* Due to the kernel treatment of Gaussian Processes, high-dimensional learning problems do not pose an issue.
  - *Trajectory learning:* Our approach presents several advantages for encoding trajectory-based policies, namely model the correlations between outputs, consider derivative relations, and also rotations.
  - *Task uncertainty:* Adopting a Heteroscedastic Gaussian Process representation we can accurately infer the latent task uncertainty from the training data. Furthermore, for time-invariant policies we develop a method for correcting the time distortions that might appear in the demonstrations, allowing us to effectively capture the variability.
  - *Task variables:* Adaptability to varying conditions is enhanced by incorporating task parameters into the model. We present a formulation for including either real, integer or categorical parameters.
  - *Modulation of the motion:* The robot can easily adapt the movement of the task by conditioning the learned policy to pass through specified via-point constraints. Additionally, several simple motions can be combined into a single, more complex one.
  - *Exploitation of replications:* We introduce a GP design that takes advantage of the structure of replications i.e., repeated demonstrations with identical conditions within data, for reducing significantly the computational cost of the learning process while retrieving an exact model.
- **Global assessment of the implications of automation in our society:** We evaluate the prospective social impacts that might come along with the development of automation from a global perspective. We analyze the possible consequences in our economy, ethics, law, and the environment, as well as provide some possible lines of action for effective management. This analysis is aimed at creating awareness among scientists and engineers, but above all, for opening the doors to reflection.

1.3 Thesis Outline

Here, we recapitulate the contents of the different chapters of the thesis and provide a reader's guide. The chapters are mostly structured to be independent of each other, in order to ease reading. At the beginning of each one, we also provide an overview of the questions addressed, so that the interested reader can skip some of the specific sections to consider a particular aspect. The thesis is organized around the following three main parts:

Part I: Theoretical Foundations

This part is aimed at those readers that are not familiar with machine learning and Gaussian Processes. It outlines the main theoretical background that is required for the comprehension of the technical aspects of the thesis. It is divided in:

Chapter 2: Machine Learning Fundamentals It starts from the fundamental concepts of nonlinear regression and probability theory, focusing also on the main properties of the Gaussian distribution, which plays a central role in this thesis. Afterward, it unifies both, regression and probability, by introducing Bayesian regression.

Chapter 3: Gaussian Processes It introduces Gaussian Processes regression, first from an intuitive point of view, and then, using the standard formulation. Thereafter, it provides an overview of the main questions regarding the model design, namely the selection of the kernel and its hyperparameters.

Part II: Robot Learning from Demonstration

The core of the thesis is encompassed in this part. It addresses the problem of easily transferring skills to robots from examples of a human teacher, presenting a complete learning from demonstration framework. It is fractionated in:

Chapter 4: Related Work It provides a comprehensive overview of the existing methods for learning robot tasks based on trajectory representations, namely, Dynamic Movement Primitives, Gaussian Mixture Models, Probabilistic Movement Primitives, and Kernelized Movement Primitives.

Chapter 5: Human to Robot Motion Transfer It presents an approach to intuitively transfer the human movements to a mobile manipulator robot using a motion capture system. The chapter reviews the correspondence problem, whole-body control, variable admittance control, and concludes with an experimental evaluation using the TIAGo robot.

Chapter 6: Learning from Demonstration with Gaussian Processes It introduces a novel Gaussian-Process-based learning framework, which encompasses the main components of a state-of-the-art method. The chapter shows how the expressiveness and versatility of Gaussian Process models can be exploited to learn trajectories, model the task uncertainty, consider task variables of different nature and adapt the learned motion. Furthermore, it presents a formulation for enhancing the scalability of Gaussian Processes when dealing with large datasets. The chapter concludes with an experimental analysis based on a door opening and a handwriting task.

Part III: Robots and Society

This part provides an overview of how the expansion of automation might shape the future of our societies. It can be read independently of the previous parts and does not require any technical background. It is divided into the following three chapters:

Chapter 7: Economic Impact of Robotics It discusses which are the prospective implications of the development of robots, from a macroeconomic perspective, and in the future of jobs. The chapter finishes with some possible solutions for managing such implications.

Chapter 8: Robot Ethics and Law The chapter introduces the rising field of roboethics and the challenges that the technological advancements pose to the current legal framework. It also concludes by presenting some potential lines of action.

Chapter 9: Environmental Impact of Robotics It addresses the environmental consequences of the evolution of robotic technologies through a systematic assessment, closing with some guidelines for an effective response.

Chapter 10: Conclusions The chapter closes the thesis providing an overview of the most relevant findings and discussing potential future research avenues. It may therefore be the starting point for the readers who wish to begin with a condensed version.

1.4 Publications

This thesis builds on the findings of the following articles, published in international journals and conferences on robotics research and artificial intelligence:

- **Miguel Arduengo**, Adrià Colomé, Júlia Borràs, Luis Sentis, and Carme Torras. Task-Adaptive Robot Learning from Demonstration with Gaussian Process Models under Replication. *IEEE Robotics and Automation Letters (RA-L)*, 6(2):966–973, 2021.
- **Miguel Arduengo**, Ana Arduengo, Adrià Colomé, Joan Lobo-Prat, and Carme Torras. Human to Robot Whole-Body Motion Transfer. *IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, Munich, Germany, July 2021.
- **Miguel Arduengo**, Adrià Colomé, Joan Lobo-Prat, Luis Sentis, and Carme Torras. Gaussian-Process-based Robot Learning from Demonstration. *ArXiv e-print*, 2020.
- **Miguel Arduengo** and Luis Sentis. The Robot Economy: Here it Comes. *International Journal of Social Robotics (SORO)*, 2020.

Part I

Theoretical Foundations

Chapter 2

Machine Learning Fundamentals

The world is waking up to the potentially transformative capabilities of machine learning, particularly when applied to robotics systems. Machine learning endows robots with learning capabilities, allowing them to move from industrial and structured environments to social and assistive domains.

Machine learning refers to a vast set of tools for understanding data. These tools can be classified as supervised or unsupervised. Broadly speaking, supervised machine learning involves building a model for predicting an output based on one or more inputs. With unsupervised machine learning, there are inputs but no supervising output; nevertheless, we can learn relationships from such data. The core objective of a learner is to generalize from its experience. In machine learning, this is done by 1) gathering a dataset, and 2) algorithmically building a statistical model based on such dataset. That statistical model is assumed to be a general representation explaining the observed data, which allows making predictions on unseen examples.

In this chapter, we examine some of the main theoretical concepts of supervised machine learning. It serves as an overview of the basic fundamentals. We only cover the tiny fraction, within the scope of interest of this thesis, of this vast field that is growing exponentially. For a more general overview, we refer the interested reader to [1], which is considered the basic reference for such purpose, and is the book on what the contents of this chapter are based on. Also, [2] provides a concise and intuitive introduction to the field.

We start, in Section 2.1, providing an intuitive example for introducing some core ideas. Afterward, in Section 2.2, we present the basic concepts of probability theory, fundamental for understanding machine learning. Then, in Section 2.3, we discuss in detail the Gaussian distribution, which plays a central role in this thesis. At the back end of this chapter, in Section 2.4, we present how to build a statistical model given a set of example input-output data from a Bayesian perspective.

2.1 Introduction: Fitting a Polynomial Curve

We begin by introducing a simple regression problem to motivate a number of key concepts. Suppose we observe a real-valued input variable x and we wish to use this observation to predict the value of a real-valued target variable y. Now, assume that we are given a training set $\mathcal{D}$ comprising N labeled example input-output pairs

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N} \tag{2.1}$$

Our goal is to exploit the training set (2.1) in order to make predictions of the value y of the target variable for some new value x of the input variable. A simple approach can be to fit the data using a polynomial function of the form

$$f(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x_j \tag{2.2}$$

where M is the order of the polynomial and the coefficients $w_0, \ldots, w_M$ are grouped in the vector $\boldsymbol{w}$. Note that, although the polynomial function $f(x, \boldsymbol{w})$ is a nonlinear function of x, it is a linear function of the coefficients $\boldsymbol{w}$. The values of the coefficients can be determined by fitting the polynomial to the training data. A possible approach to do so is to minimize an error function that measures the misfit between the function $f(x, \boldsymbol{w})$, for any given value of $\boldsymbol{w}$, and the training data. One simple choice of the error function is the sum-of-squares, which is expressed as

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{i=1}^{N} \left(f(x_i, \boldsymbol{w}) - y_i\right)^2 \tag{2.3}$$

where the factor of $1/2$ is included for later convenience. Note that it is a non-negative function whose minimum is reached if function $f(x, \boldsymbol{w})$ passes exactly through each training point. We can also write (2.3) in matrix form as

$$E(\boldsymbol{w}) = \frac{1}{2} \left(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\right)^T \left(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\right) \tag{2.4}$$

where the matrix $\boldsymbol{X}$ and the vector $\boldsymbol{y}$ refer to

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1 & \ldots & x_1^M \\ 1 & x_2 & \ldots & x_2^M \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & \ldots & x_N^M \end{bmatrix}, \qquad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \tag{2.5}$$

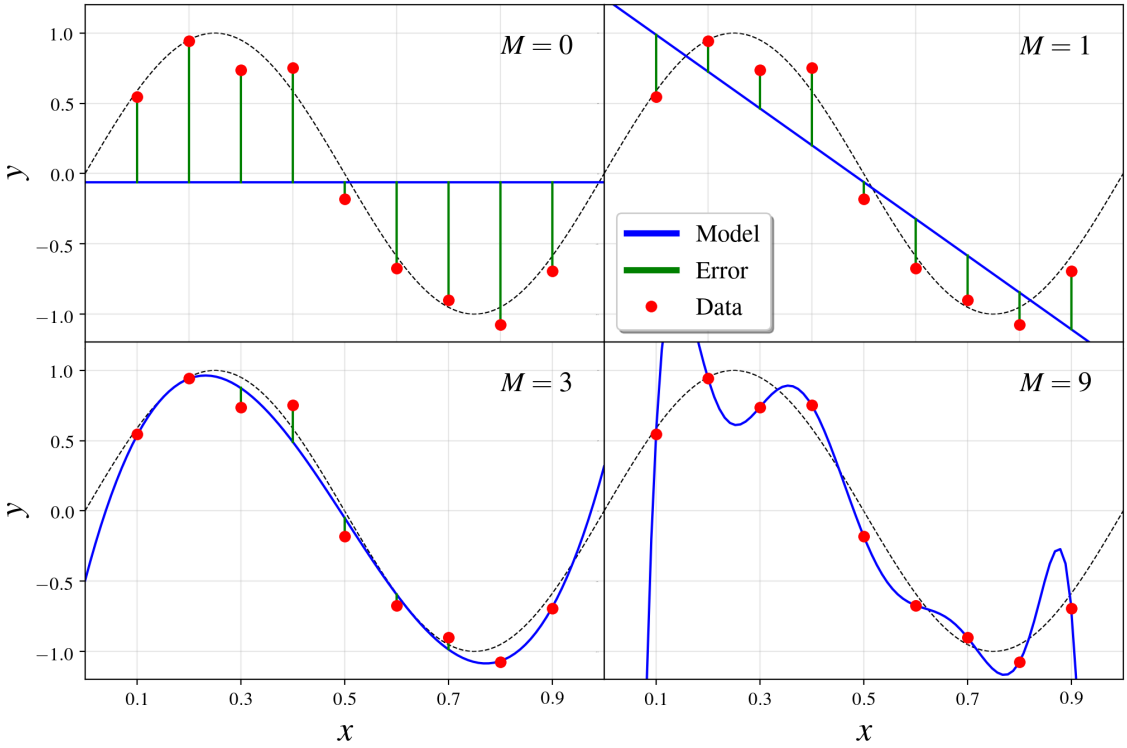

Figure 2.1: Polynomial models of different orders M fitted to the training data. The training data are noisy samples from the function represented as a black dashed line.

To find the optimal coefficients $\boldsymbol{w}^*$ that minimize the error (2.4), we can derive with respect to $\boldsymbol{w}$ and equate to zero obtaining

$$\frac{\partial E(\boldsymbol{w})}{\partial \boldsymbol{w}} = \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y} \longrightarrow \boldsymbol{w}^* = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{2.6}$$

Equation (2.6) is the well-known least-squares formula. We can see that since the error function (2.4) is a quadratic function of the coefficients $\boldsymbol{w}$, its derivatives with respect to the coefficients are linear in the elements of $\boldsymbol{w}$, and so the minimization of the error function has a unique solution which can be written in closed form.

There remains the problem of choosing the order M of the polynomial. This is related to the concept of model selection. For illustrating it, consider the example shown in Figure 2.1. We can see that the constant ($M = 0$) and first-order ($M = 1$) polynomials result in a poor fit to the training data and consequently they are poor models. In this case, we say that we are underfitting the data. For the high order polynomial ($M = 9$) the model passes exactly through the training points but it does not generalize well on new input points due to the large oscillations. In this case, we say that we are overfitting the data. We have that the third-order ($M = 3$) polynomial retrieves the best model among the considered ones.

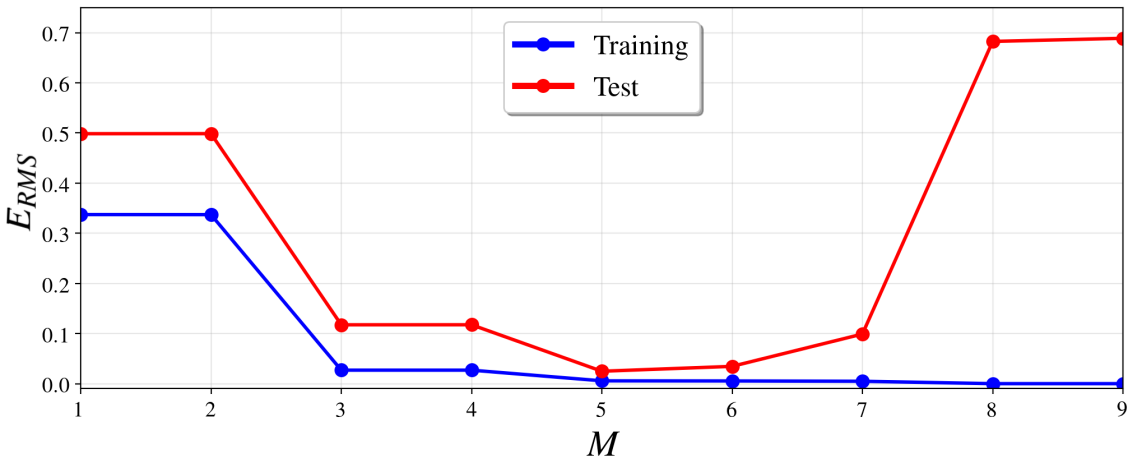

Figure 2.2: Root mean square error (2.7) evaluated on the training set in Figure 2.1 and on an independent test set for different orders M of the fitting polynomial.

In machine learning, the goal is to obtain a model with good generalization capabilities i.e. able to make accurate predictions for unseen data. We can obtain some quantitative insight into the dependence of the generalization performance on M by considering a separate test dataset. For each choice of M, we can then evaluate the residual value of $E(\boldsymbol{w}^*)$ for the training and test datasets. In this case, is more convenient to use the root-mean-square (RMS) error defined by

$$E_{\mathrm{RMS}} = \sqrt{\frac{2E(\boldsymbol{w}^*)}{N}} \tag{2.7}$$

in which the division by N allows us to compare the error between datasets of different sizes. In Figure 2.2 we show the RMS over the training and test sets for several values of the fitting polynomial order M. The training error measures the fit to the training data while the test set error measures the accuracy of the predictions for unseen data. We can see that the training error decreases as the order of the polynomial increases, going to zero for $M = 9$, as we can verify in Figure 2.1. Regarding the test error, we can see that is relatively low in the range $3 \leq M \leq 7$ and relatively high otherwise. For $M < 3$ we are underfitting to the training data because the model is too simple, while for $M > 7$ we are overfitting to the data because the model is too complex. The best generalization capabilities are achieved for $M = 5$, where the test error reaches its minimum.

Additionally, it is also interesting to examine the behavior of the resulting model as the size of the training dataset is varied. In Figure 2.3 we can see that, for a given model complexity, the overfitting problem becomes less severe as the size of the dataset increases. Another way to say this is that the larger the data set, the more complex the model that we can afford to fit the data.

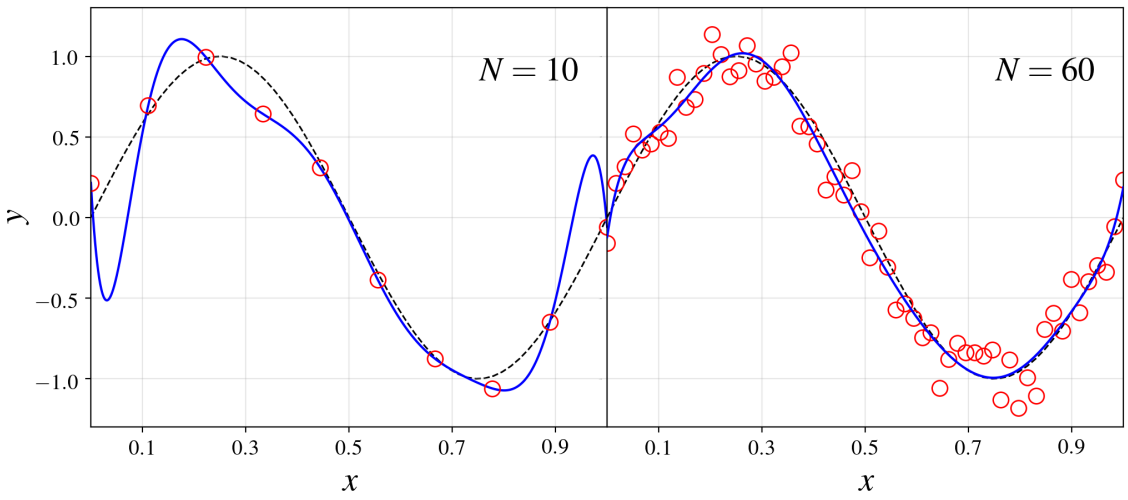

Figure 2.3: Polynomial fit of degree $M = 9$ that minimizes the sum-of-square errors (2.3). Increasing the size of the training dataset N the overfitting is reduced.

The question now is how can we fit a model to a dataset of limited size when we want to use relatively complex and flexible models in practice. One technique that is often used to control the overfitting problem is that of regularization, which involves including a penalty term to the error function (2.3) in order to discourage the coefficients from reaching large values

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}(f(x_i, \boldsymbol{w}) - y_i)^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \tag{2.8}$$

where the coefficient λ weights the relative importance of the regularization term compared with the sum-of-squares error. Again, the error function (2.8) can be minimized in closed form, resulting in the optimal weights

$$\boldsymbol{w}^* = \left(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{2.9}$$

being $\boldsymbol{I}$ the identity matrix. This type of regression is commonly known as ridge regression. In Figure 2.4, we show the results of fitting a polynomial of order $M = 9$ to the same training data of Figure 2.1 using the regularized error function (2.8). We can see that the coefficient λ effectively controls the complexity of the resulting model and hence determines the degree of overfitting. The problem now is how to determine the best value of λ. A simple approach for achieving this is by partitioning the available data in a training set, used to determine $\boldsymbol{w}$, and a separate validation set, used to determine the best model parameters (M and λ). If the available data is limited, we have to seek more sophisticated methods such as cross-validation [3] or Akaike's Information Criterion [4].

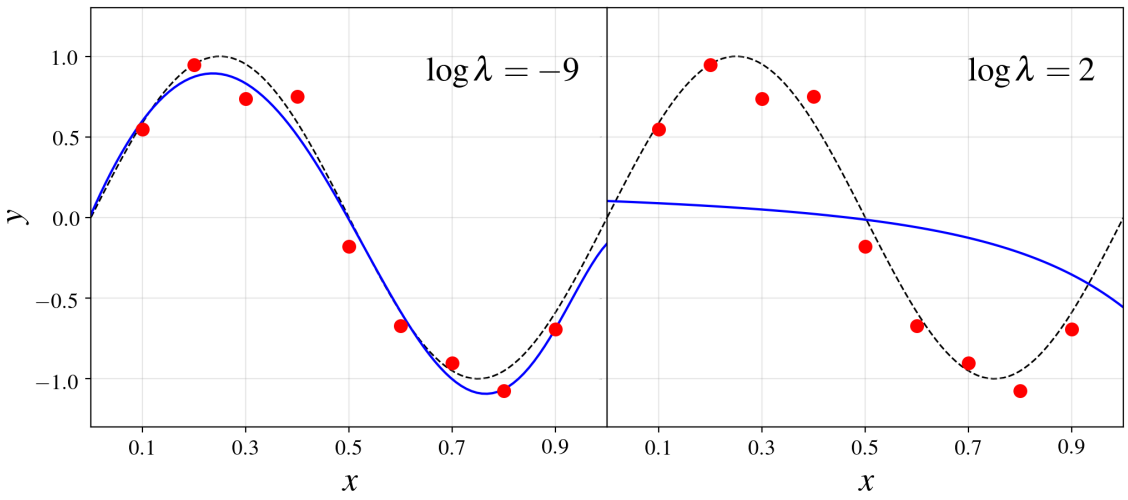

Figure 2.4: Polynomial fit of degree $M = 9$ that minimizes the regularized error function (2.8) for different values of the coefficient λ.

2.2 Probability Theory

In this section, we seek a mathematical formulation that allows us to address machine learning problems in a more formal and systematic way. The solution can be found resorting to probability theory. The key link between machine learning and probability lies in the concept of uncertainty. It arises both through the noise on measurements, as well as through the finite size of datasets. Probability theory provides a consistent framework for handling such uncertainty and constitutes one of the central pillars of machine learning.

The basic concepts of probability can be illustrated through a simple example, illustrated in Figure 2.5. Imagine that we have two boxes of different colors, one green and one red, each one containing a different number of black and white balls. Now, imagine we randomly pick a ball from one of the boxes, and having observed its color, we put it back in the box. This process is repeated many times choosing the green box 40 percent of the time and the red box the remaining 60 percent. When we pick a ball, we are equally likely to pick any of the balls in the box. The box is analogous to a random variable B. This random variable can take two possible values, namely g (green) or r (red). Similarly, the ball is also a random variable L, which can take either value b (black) or w (white).

The probability of an event can be interpreted as the fraction of times that event occurs out of the total number of experiments when such a number goes to infinity. Thus, the probability of selecting the green box is $p(B = g) = 4/10$ and the probability of selecting the red box is $p(B = r) = 6/10$. By definition, probabilities must lie in the interval $[0, 1]$. Also, if the events are mutually exclusive and include all possible outcomes, the probabilities for those events must sum to one.

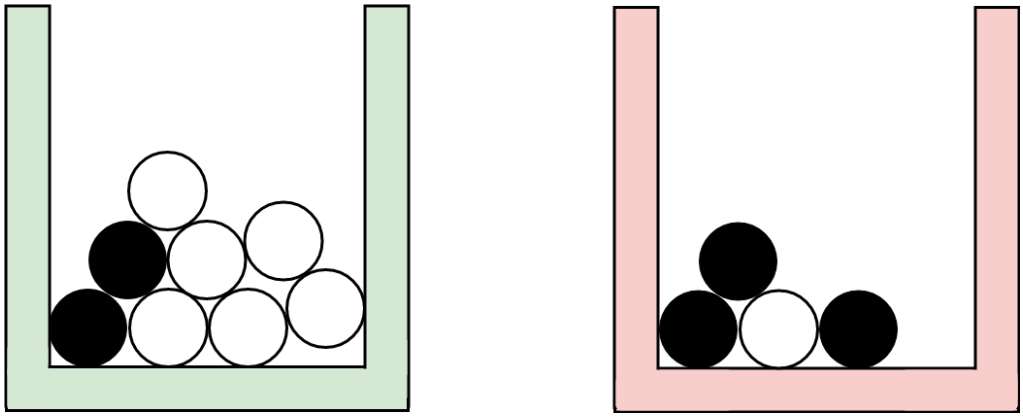

Figure 2.5: In the black box, on the left, we have 2 black balls and 6 white balls. In the red box, on the right, we have 3 black balls and 1 white ball.

We could now ask ourselves, what is the overall probability of picking a white ball? Intuitively, in Figure 2.5, we can see that given that we pick the green box, the probability is 6/8. On the other hand, given that we pick the red box, the probability is 1/4. To compute the overall probability, we have to consider that we are picking more times the red box than the green box, thus it should be closer to 1/4 than 6/8. How much closer? We can weigh it depending on the probability of picking each type of box. That is,

$$p(L = w) = p(L = w \mid B = g)p(B = g) + p(L = w \mid B = r)p(B = r) = 9/20 \quad (2.10)$$

where $\mid$ is equivalent to 'given that'. Implicitly, in (2.10), we have used the two fundamental rules of probability theory: the sum rule and the product rule. For a more general case, let X and Y be two random variables, where X can take any of the values x_i, for $i = 1, \ldots, N$, and Y can take take any of the values y_j, for $j = 1, \ldots, M$. The aforementioned rules can then be written as

$$\textbf{Sum rule}: \quad p(X = x_i) = \sum_{j=1}^{M} p(X = X_i, Y = y_j) \quad (2.11)$$

$$\textbf{Product rule}: \quad p(X = x_i, Y = y_j) = p(Y = y_j \mid X = x_i)p(X = x_i) \quad (2.12)$$

For the sake of simplicity, from now on we denote $p(X) \equiv p(X = x_i)$. Here $p(X, Y)$ is known as the joint probability, and $p(Y \mid X)$ as the conditional probability. Another question we may ask ourselves is, given that we have chosen a white ball, what is the probability that the box we selected was the red one? To answer it we have to resort to the Bayes' theorem, which can be written as

$$p(Y \mid X) = \frac{p(X \mid Y)p(Y)}{p(X)} \tag{2.13}$$

It is straightforward to derive from the product rule (2.12) together with the symmetry property $p(X, Y) = p(Y, X)$. Using the sum rule we can rewrite the denominator in (2.13) as

$$p(X) = \sum_{Y} p(X \mid Y)p(Y) \tag{2.14}$$

Then, substituting (2.14) in (2.13), we have that the probability that the selected box is the red one, given that we have chosen a white ball is

$$p(B = r \mid L = w) = \frac{1/4 \cdot 6/10}{1/4 \cdot 6/10 + 6/8 \cdot 4/10} = \frac{1}{3} \tag{2.15}$$

If we had been asked which box had been chosen before being told the color of the picked ball, then the most complete information we have available is $p(B = r)$. We call this the prior probability because it is the probability available before. Once we are told that the ball is white, we can then use Bayes' theorem to compute the probability $p(B = r \mid L = w)$, which is known as the posterior probability because it is the probability obtained after we have observed L. In the presented example the prior probability of selecting the red box is 6/10. However, once we observe the color of the ball it is more likely that we have picked the green box since the posterior probability of the red box is 1/3. This result is intuitive, since the proportion of white balls is much higher in the green box, and such observation provides significant evidence favoring the green box.

Finally, if each of the boxes in Figure 2.5 contained the same fraction of black and white balls we would have that the probability of picking a white ball is independent of which box we pick. Then, B and L are said to be independent. If X and Y are independent, they fulfill the following identities:

$$p(X, Y) = p(X)p(Y) \tag{2.16}$$
$$p(X \mid Y) = p(X) \tag{2.17}$$

In the remainder of this section we extend our discussion on probability theory based on the introduced core concepts. First, in Section 2.2.1, we present how to jump from discrete random variables to continuous ones. Then, in Section 2.2.2, we introduce the definition of expectation and covariance. Finally, in Section 2.2.3 we briefly discuss the Bayesian interpretation of probability.

2.2.1 Probability Densities

Random variables are not always discrete, in some cases, they can take an infinite number of different values in a certain range. The probability density function allows extending the concepts previously presented for discrete events to continuous variables. Let $p(x)$ be the probability density over the real-valued variable x. The probability that x lies in the interval (a, b) is given by

$$p(a \leq x \leq b) = \int_a^b p(x)dx \tag{2.18}$$

Because probabilities are non-negative, and the value of x must lie in the real axis, the probability density $p(x)$ must satisfy the following conditions

$$p(x) \geq 0 \tag{2.19}$$

$$\int_{-\infty}^{\infty} p(x)dx = 1 \tag{2.20}$$

The probability that x lies in the interval $(-\infty, z)$ is given by the cumulative distribution function defined by

$$P(z) = \int_{-\infty}^{z} p(x)dx \tag{2.21}$$

Finally, the sum and product rules of probability, as well as Bayes' theorem, apply equally to the case of probability densities

$$\textbf{Sum rule}: \quad p(x) = \int p(x, y)dy \tag{2.22}$$

$$\textbf{Product rule}: \quad p(x, y) = p(y \mid x)p(x) \tag{2.23}$$

$$\textbf{Bayes' theorem}: \quad p(y \mid x) = \frac{p(y \mid x)p(y)}{\int p(x, y)p(y)dy} \tag{2.24}$$

where both x and y are real variables. If we have several continuous variables $x_1, \ldots, x_D$ grouped in a vector $\boldsymbol{x}$, the aforementioned definitions are identical. In Figure 2.6 we show an example of a probability density function $p(x)$ and the corresponding cumulative distribution $P(x)$. We can see that since $p(x) \geq 0$, the cumulative distribution is an increasing function.

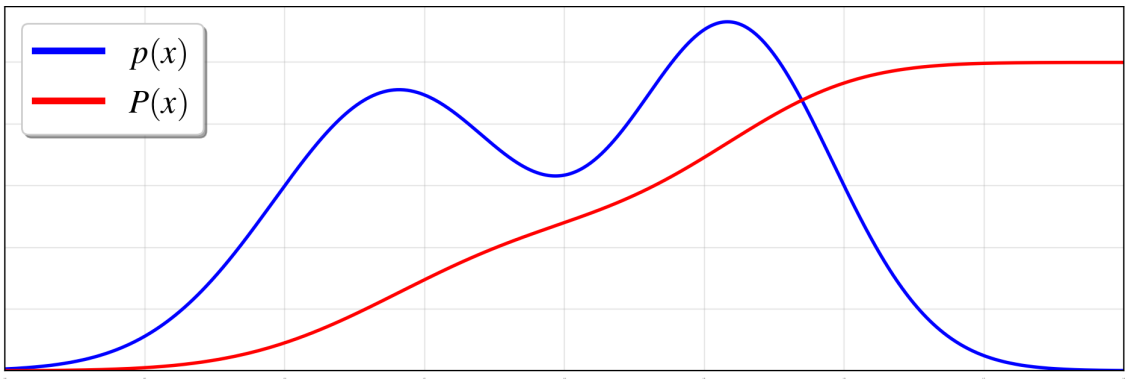

Figure 2.6: Probability density function $p(x)$ (2.18) along with the corresponding cumulative distribution $P(x)$ (2.21). Note that $p(x)$ is the derivative of $P(x)$.

2.2.2 Expectation and Covariance

One of the most important operations involving probabilities is that of finding the expected value of a function $f(x)$. This operation can be seen as a weighted average of the function, being the relative weights determined by the probability of taking such value. For a continuous variable x the expectation of $f(x)$ is given by

$$\mathbb{E}\left[f(x)\right] = \int f(x)p(x)dx \tag{2.25}$$

We can also consider a conditional expectation with respect to a conditional distribution, so that

$$\mathbb{E}\left[f(x) \mid y\right] = \int f(x)p(x \mid y)dx \tag{2.26}$$

The variance of $f(x)$ provides a measure of how much variability there is in $f(x)$ around its mean value $\mathbb{E}\left[f(x)\right]$

$$\mathbb{V}\left[f(x)\right] = \mathbb{E}\left[\left(f(x) - \mathbb{E}\left[f(x)\right]\right)^2\right] = \mathbb{E}\left[f(x)^2\right] - \mathbb{E}\left[f(x)\right]^2 \tag{2.27}$$

Finally, for two random variables x and y, the covariance expresses the extent to which x and y are related to each other. It is defined by

$$\operatorname{cov}\left[x, y\right] = \mathbb{E}\left[x \cdot y\right] - \mathbb{E}\left[x\right]\mathbb{E}\left[y\right] \tag{2.28}$$

If x and y are independent $\mathbb{E}\left[x \cdot y\right] = \mathbb{E}\left[x\right]\mathbb{E}\left[y\right]$. Then, their covariance vanishes

$$\operatorname{cov}\left[x, y\right] = 0 \tag{2.29}$$

2.2.3 Bayesian Probability

Intuitively, probabilities can be seen in terms of the frequencies of random, repeatable events. This is known in the literature as the classical or frequentist interpretation of probability. A more interesting interpretation in machine learning is the Bayesian one, in which probabilities provide a measure of the uncertainty. Consider an uncertain event, for instance, whether there will be a robot in every home by the end of the century. This is not an event that can be repeated numerous times in order to define a notion of probability. Nevertheless, we can have some idea, for example, of how fast are robotics technologies being developed. Our assessment of such matters will affect the actions we take, for instance, the extent to which we endeavor to write the laws for regulating the use of robots. Through the Bayesian interpretation of probability, we can estimate the uncertainty of such events and make revisions to them in the light of new evidence.

Bayes' theorem (2.13) now acquires a new meaning. For instance, consider the example of inferring the parameters $\boldsymbol{w}$ in polynomial curve fitting (2.2). We can capture our assumptions about $\boldsymbol{w}$, before observing any data, in the form of a *prior* distribution $p(\boldsymbol{w})$. Then, the effect of the training data $\mathcal{D}$ (2.1) is expressed through the conditional probability $p(\mathcal{D} \mid \boldsymbol{w})$. Bayes' theorem now takes the form

$$p(\boldsymbol{w} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{w})p(\boldsymbol{w})}{p(\mathcal{D})} \tag{2.30}$$

This allows us to evaluate the uncertainty in $\boldsymbol{w}$ after we have observed $\mathcal{D}$ in the form of the *posterior* probability $p(\boldsymbol{w} \mid \mathcal{D})$. The term $p(\mathcal{D} \mid \boldsymbol{w})$ in (2.30) is known as the *likelihood* function. It expresses how probable the observed data set is for different settings of the parameter vector $\boldsymbol{w}$. Given this definition, we can also state Bayes' theorem in words as

$$\text{posterior} \propto \text{likelihood} \times \text{prior} \tag{2.31}$$

where all of these quantities are viewed as functions of $\boldsymbol{w}$. From this perspective, two alternative methods can be used to estimate $\boldsymbol{w}$. If we do not have prior information available, we can use maximum likelihood estimation (MLE)

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} p(\mathcal{D} \mid \boldsymbol{w}) \tag{2.32}$$

On the other hand, if we have prior knowledge at hand, we can use maximum a-posteriori probability (MAP) estimation

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w} \mid \mathcal{D}) \tag{2.33}$$

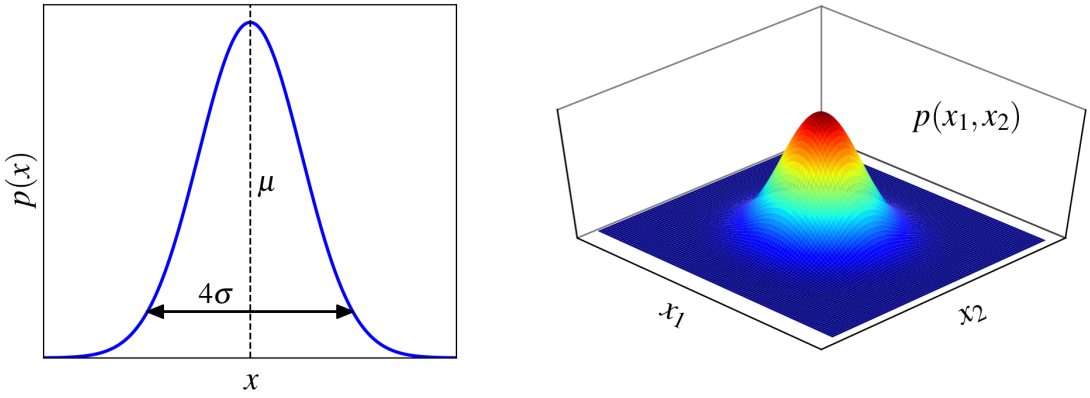

Figure 2.7: On the left, Gaussian distribution for a one-dimensional variable (2.34). On the right for a two-dimensional variable (2.35).

2.3 The Gaussian Distribution

The Gaussian distribution, also known as the normal distribution, is a widely used model for continuous variables. In the case of a single variable x, the Gaussian distribution can be written in the form

$$\mathcal{N}\left(\mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{\sigma^2}\right) \tag{2.34}$$

where μ is the mean and σ^2 is the variance. For a D-dimensional vector $\boldsymbol{x}$, the Gaussian distribution takes the form

$$\mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = \frac{1}{\sqrt{(2\pi)^D \left|\boldsymbol{\Sigma}\right|}} \exp\left(-\frac{1}{2}\left(\boldsymbol{x}-\boldsymbol{\mu}\right)^T \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu}\right)\right) \tag{2.35}$$

where $\boldsymbol{\mu}$ is a D-dimensional vector, $\boldsymbol{\Sigma}$ is a symmetric $D \times D$ matrix, and $|\boldsymbol{\Sigma}|$ denotes the determinant of $\boldsymbol{\Sigma}$. In Figure 2.7, we can see a graphical representation of the Gaussian distribution. From a theoretical perspective the interpretation of the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ is given by the mean and the covariance

$$\mathbb{E}\left[\boldsymbol{x}\right] = \boldsymbol{\mu}, \qquad \operatorname{cov}\left[\boldsymbol{x}, \boldsymbol{x}\right] = \boldsymbol{\Sigma} \tag{2.36}$$

In this section, we provide an overview of the main aspects of Gaussian distributions, which commonly play a central role in probabilistic frameworks. We derive, in Section 2.3.1, the conditional Gaussian distribution, and in Section 2.3.2, the marginal Gaussian distribution.

2.3.1 Conditional Gaussian Distribution

An important property of the multivariate Gaussian distribution is that if two variables are jointly Gaussian, then the conditional distribution of one conditioned on the other is again Gaussian. Let $\boldsymbol{x}$ be a D-dimensional random variable normally distributed according to $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Without loss of generality, we can partition $\boldsymbol{x}$ in two components $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, comprising the former the first M components and the latter the remaining $D - M$ components, so that

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} \tag{2.37}$$

We also define the corresponding partitions of the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \qquad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix} \tag{2.38}$$

Now, it is convenient to work with the inverse of the covariance matrix

$$\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1} \longrightarrow \begin{bmatrix} \boldsymbol{\Lambda}_{11} & \boldsymbol{\Lambda}_{12} \\ \boldsymbol{\Lambda}_{21} & \boldsymbol{\Lambda}_{22} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}^{-1} \tag{2.39}$$

Intuitively, finding the conditional probability $p(\boldsymbol{x}_1 \mid \boldsymbol{x}_2)$ is equivalent to compute the probability $p(\boldsymbol{x}_1, \boldsymbol{x}_2)$ regarding $\boldsymbol{x}_2$ as a constant. Using the decomposition (2.38) we can rewrite the exponent in the Gaussian distribution (2.35) as

$$\begin{aligned} &-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) = \\ &-\frac{1}{2}(\boldsymbol{x}_1-\boldsymbol{\mu}_1)^T \boldsymbol{\Lambda}_{11}(\boldsymbol{x}_1-\boldsymbol{\mu}_1) - \frac{1}{2}(\boldsymbol{x}_1-\boldsymbol{\mu}_1)^T \boldsymbol{\Lambda}_{12}(\boldsymbol{x}_2-\boldsymbol{\mu}_2) \\ &-\frac{1}{2}(\boldsymbol{x}_2-\boldsymbol{\mu}_2)^T \boldsymbol{\Lambda}_{21}(\boldsymbol{x}_1-\boldsymbol{\mu}_1) - \frac{1}{2}(\boldsymbol{x}_2-\boldsymbol{\mu}_2)^T \boldsymbol{\Lambda}_{22}(\boldsymbol{x}_2-\boldsymbol{\mu}_2) \end{aligned} \tag{2.40}$$

The first thing that we should note is that as for the original Gaussian distribution for $\boldsymbol{x}$, this is also a quadratic form as a function of $\boldsymbol{x}_1$, and hence the corresponding conditional distribution $p(\boldsymbol{x}_1 \mid \boldsymbol{x}_2)$ is also Gaussian. Our goal is to find the mean $\boldsymbol{\mu}_{1|2}$ and covariance of $\boldsymbol{\Sigma}_{1|2}$ such new Gaussian distribution. We can achieve this by noting that the exponent in general Gaussian distribution, considering the symmetry of $\boldsymbol{\Sigma}$, can be written as

$$-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) = -\frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \text{constant terms} \tag{2.41}$$

Comparing (2.40) and (2.41), separating all the terms that are second and first order in $\boldsymbol{x}_1$ we obtain

$$\begin{aligned} -\frac{1}{2}\left(\boldsymbol{x}-\boldsymbol{\mu}\right)^T \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu}\right) = &-\frac{1}{2}\boldsymbol{x}_1^T \boldsymbol{\Lambda}_{11}\boldsymbol{x}_1 \\ &- \boldsymbol{x}_1^T\left(\boldsymbol{\Lambda}_{11}\boldsymbol{\mu}_1 - \boldsymbol{\Lambda}_{12}\left(\boldsymbol{x}_2-\boldsymbol{\mu}_2\right)\right) + \text{cte} \end{aligned} \tag{2.42}$$

where we have used $\boldsymbol{\Lambda}_{12}^T = \boldsymbol{\Lambda}_{21}$. Then, it is immediate to see that the mean $\boldsymbol{\mu}_{1|2}$ and covariance $\boldsymbol{\Sigma}_{1|2}$ of the conditional distribution are equivalent to

$$\boldsymbol{\Sigma}_{1|2} = \boldsymbol{\Lambda}_{11}^{-1} \tag{2.43}$$

$$\boldsymbol{\mu}_{1|2} = \boldsymbol{\Sigma}_{1|2}\left(\boldsymbol{\Lambda}_{11}\boldsymbol{\mu}_1 - \boldsymbol{\Lambda}_{12}\left(\boldsymbol{x}_2-\boldsymbol{\mu}_2\right)\right) = \boldsymbol{\mu}_1 - \boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\Lambda}_{12}\left(\boldsymbol{x}_2-\boldsymbol{\mu}_2\right) \tag{2.44}$$

We can also express equations (2.42) and (2.43) in terms of the original covariance matrix $\boldsymbol{\Sigma}$ using the following matrix identity

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{M} & -\boldsymbol{M}\boldsymbol{B}\boldsymbol{D}^{-1} \\ -\boldsymbol{D}^{-1}\boldsymbol{C}\boldsymbol{M} & \boldsymbol{D}^{-1} + \boldsymbol{D}^{-1}\boldsymbol{C}\boldsymbol{M}\boldsymbol{B}\boldsymbol{D}^{-1} \end{bmatrix} \tag{2.45}$$

being the matrix $\boldsymbol{M}$ expressed as

$$\boldsymbol{M} = \left(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}\right)^{-1} \tag{2.46}$$

Then, making use of (2.45) we obtain the following equivalences

$$\boldsymbol{\Lambda}_{11} = \left(\boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}\right)^{-1} \tag{2.47}$$

$$\boldsymbol{\Lambda}_{12} = -\left(\boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}\right)^{-1}\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1} \tag{2.48}$$

Finally, substituting (2.47) and (2.48), in (2.43) and (2.44), yields the following mean $\boldsymbol{\mu}_{1|2}$ and covariance $\boldsymbol{\Sigma}_{1|2}$ for the conditional distribution $p(x_2 \mid x_1)$

$$\boldsymbol{\mu}_{1|2} = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\left(\boldsymbol{x}_2-\boldsymbol{\mu}_2\right) \tag{2.49}$$

$$\boldsymbol{\Sigma}_{1|2} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} \tag{2.50}$$

An example of the conditional probability distribution for a Gaussian over two variables is shown in Figure 2.8.

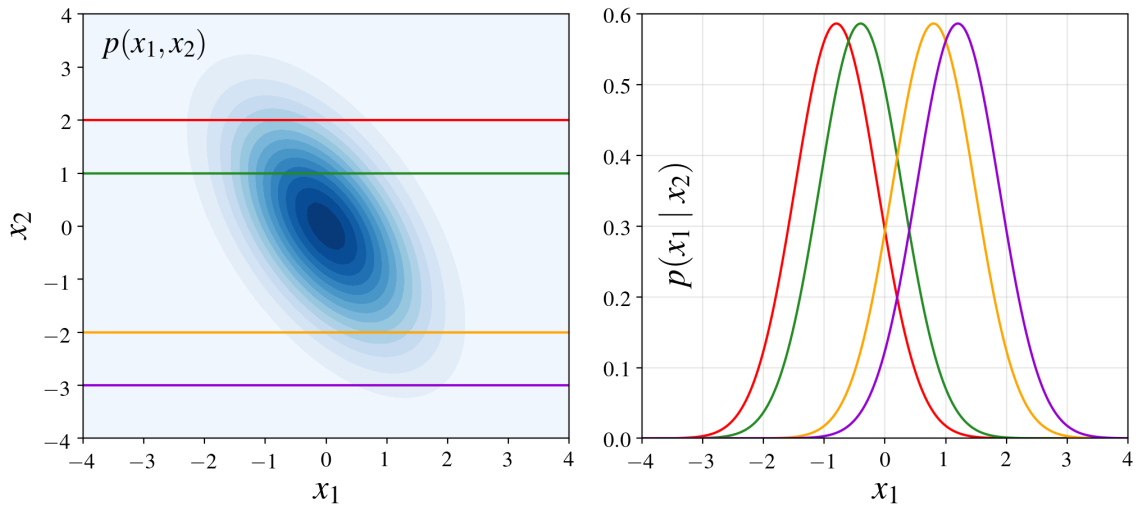

Figure 2.8: On the left, contour plot of a Gaussian distribution over two variables $p(x_1, x_2)$. On the right, the corresponding conditional probability distribution $p(x_1 \mid x_2)$ for different values of x_2, represented as horizontal lines with the same colors in the left plot.

2.3.2 Marginal Gaussian Distribution

Now, we turn to a discussion of the marginal Gaussian distribution. Using the sum rule (2.22), it is given by

$$p(\boldsymbol{x}_2) = \int p(\boldsymbol{x}_1, \boldsymbol{x}_2) d\boldsymbol{x}_1 \tag{2.51}$$

Similarly to the conditional distribution, if $p(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is Gaussian, $p(\boldsymbol{x}_2)$ will be Gaussian again. Once again, our strategy for evaluating this distribution is to focus on the quadratic form in the exponent of the joint distribution and thereby identify the mean and covariance. Per definition, the exponent of the joint Gaussian distribution, using the decomposition (2.38), can be written as

$$-\frac{1}{2}\left(\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix} - \begin{bmatrix}\boldsymbol{\mu}_1\\ \boldsymbol{\mu}_2\end{bmatrix}\right)^T \begin{bmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12}\\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22}\end{bmatrix}^{-1} \left(\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix} - \begin{bmatrix}\boldsymbol{\mu}_1\\ \boldsymbol{\mu}_2\end{bmatrix}\right) \tag{2.52}$$

For the matrix inverse we can use (2.45) and (2.46). Defining $\widetilde{\boldsymbol{x}}_i = \boldsymbol{x}_i - \boldsymbol{\mu}_i$, we can now rewrite (2.52) as

$$-\frac{1}{2}\begin{bmatrix}\widetilde{\boldsymbol{x}}_1\\ \widetilde{\boldsymbol{x}}_2\end{bmatrix}^T \begin{bmatrix}\boldsymbol{\Lambda}_{11} & -\boldsymbol{\Lambda}_{11}\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\\ -\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}\boldsymbol{\Lambda}_{11} & \boldsymbol{\Sigma}_{22}^{-1} + \boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}\boldsymbol{\Lambda}_{11}\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\end{bmatrix} \begin{bmatrix}\widetilde{\boldsymbol{x}}_1\\ \widetilde{\boldsymbol{x}}_2\end{bmatrix} \tag{2.53}$$

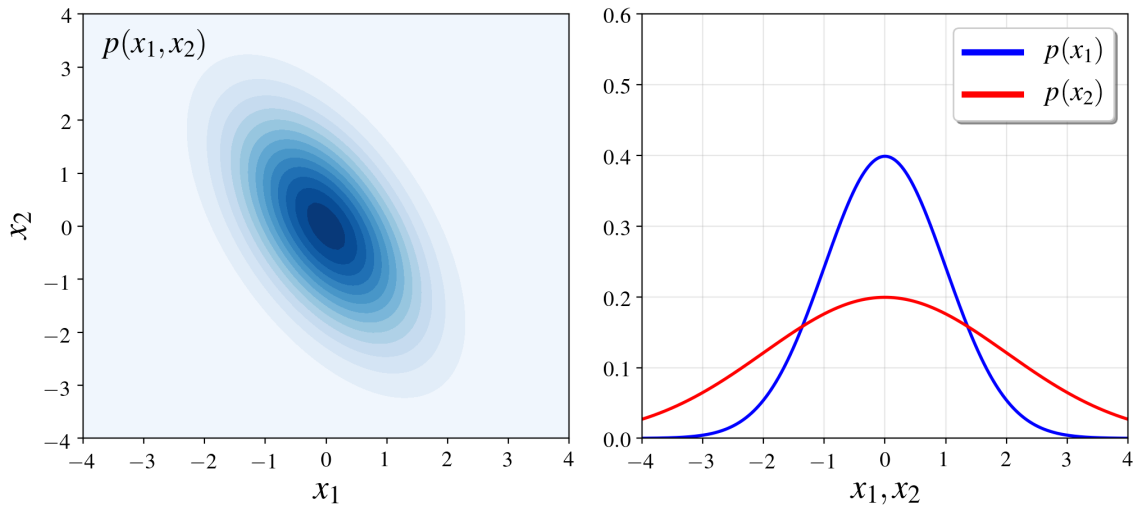

Figure 2.9: On the left, contour plot of a Gaussian distribution over two variables $p(x_1, x_2)$. On the right, the corresponding marginal probability distributions $p(x_1)$ and $p(x_2)$.

Then, we can split up (2.53) into two separate exponents, through

$$-\frac{1}{2}\widetilde{\boldsymbol{x}}_2^T\boldsymbol{\Sigma}_{22}^{-1}\widetilde{\boldsymbol{x}}_2 - \frac{1}{2}\begin{bmatrix}\widetilde{\boldsymbol{x}}_1\\ \widetilde{\boldsymbol{x}}_2\end{bmatrix}^T\begin{bmatrix}-\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} & \boldsymbol{I}\end{bmatrix}\boldsymbol{\Lambda}_{11}\begin{bmatrix}-\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\\ \boldsymbol{I}\end{bmatrix}\begin{bmatrix}\widetilde{\boldsymbol{x}}_1\\ \widetilde{\boldsymbol{x}}_2\end{bmatrix} \tag{2.54}$$

Noting that $\boldsymbol{\Sigma}_{12} = \boldsymbol{\Sigma}_{21}^T$ and working the right half, we can find that (2.54) equals

$$-\frac{1}{2}\widetilde{\boldsymbol{x}}_2^T\boldsymbol{\Sigma}_{22}^{-1}\widetilde{\boldsymbol{x}}_2 - \frac{1}{2}\left(\widetilde{\boldsymbol{x}}_1 - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\widetilde{\boldsymbol{x}}_2\right)^T\boldsymbol{\Lambda}_{11}\left(\widetilde{\boldsymbol{x}}_1 - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\widetilde{\boldsymbol{x}}_2\right) \tag{2.55}$$

By inspection of (2.55), recalling the expression of $\boldsymbol{\Lambda}_{11}$ (2.47), the conditional mean (2.49) and covariance (2.50), it is immediate that

$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}\left(\boldsymbol{\mu}_{1|2}, \boldsymbol{\Sigma}_{1|2}\right)\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_{22}) \tag{2.56}$$

From the product rule (2.23), we now that $p(\boldsymbol{x}_1, \boldsymbol{x}_2) = p(\boldsymbol{x}_1 \mid \boldsymbol{x}_2)p(\boldsymbol{x}_2)$. Thus, it follows that the marginal distribution is given by

$$p(\boldsymbol{x}_2) = \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_{22}) \tag{2.57}$$

Intuitively, we see that to obtain the marginal distribution over a subset of multivariate normal random variables, one only needs to drop the irrelevant variables (the variables that one wants to marginalize out) from the partitioned mean vector and the covariance matrix. An example of the marginal distribution associated with a two-variable multivariate Gaussian is shown in Figure 2.9.

2.4 Bayesian Linear Regression

The goal of regression is to predict the value of a continuous target variable y given the value of a D-dimensional vector $\boldsymbol{x}$ of input variables. We have already encountered an example of a regression problem when we considered polynomial curve fitting in Section 2.1. The polynomial is a specific example of a broad class of functions called linear regression models, which share the property of being linear functions of the model parameters $\boldsymbol{w}$.

From a Bayesian perspective, a regression can be seen as a model of the predictive distribution $p(y \mid \boldsymbol{x})$ because this expresses our uncertainty about the estimated value of y for each value of $\boldsymbol{x}$. From this conditional distribution, we can make predictions of y, for any new value of $\boldsymbol{x}$, in such a way as to minimize the expected value of a suitably chosen loss function.

In this section, we return to the curve-fitting problem, approaching it from a probabilistic perspective, thereby gaining some insights into error functions and regularization, as well as taking us towards a full Bayesian treatment. We start, in Section 2.4.1, presenting the linear regression model that we adopt throughout the section. Afterward, in Section 2.4.2, we discuss the model complexity problem using probabilistic formalism. Then, in Sections 2.4.3 and 2.4.4, we show a Bayesian approach to compute the model parameters $\boldsymbol{w}$ and make predictions for new inputs $\boldsymbol{x}$, respectively. Finally, in Section 2.4.5, we present the analogy between Bayesian regression and kernel methods, which play a central role in this thesis.

2.4.1 Linear Basis Functions Models

The simplest linear model for regression involves a linear combination of the input variables $\boldsymbol{x}$ and the model parameters $\boldsymbol{w}$

$$f(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1 x_1 + \cdots + w_D x_D = \boldsymbol{w}^T \boldsymbol{x} \tag{2.58}$$

The key property of this model is that it is a linear function of the parameters $\boldsymbol{w}$. It is also, however, a linear function of the input variables x_i, which imposes significant limitations on the model. To address this issue, we can extend the class of models by considering linear combinations of fixed nonlinear functions of the input variables, of the form

$$f(\boldsymbol{x}, \boldsymbol{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) \tag{2.59}$$

where $\phi_j(\boldsymbol{x})$ are known as basis functions, M is the dimension of $\boldsymbol{w}$ and we have taken $\phi_0(\boldsymbol{x}) = 1$. By using nonlinear basis functions, we allow the function $f(\boldsymbol{x}, \boldsymbol{w})$ to be a nonlinear function of the input vector $\boldsymbol{x}$. There are many possible choices for the basis functions, a common one is

$$\phi_j(\boldsymbol{x}) = \exp\left(-\frac{1}{2}\frac{(\boldsymbol{x} - \boldsymbol{\mu}_j)^T(\boldsymbol{x} - \boldsymbol{\mu}_j)}{\ell^2}\right) \tag{2.60}$$

where the $\boldsymbol{\mu}_j$ governs the locations of the basis functions in the input space and ℓ their spatial scale. These are usually referred to as Gaussian or radial basis functions. Now, let us assume that the observations of the output y are given by the deterministic function $f(\boldsymbol{x}, \boldsymbol{w})$ affected by additive Gaussian noise so that

$$y = f(\boldsymbol{x}, \boldsymbol{w}) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}\left(0, \sigma_n^2\right) \tag{2.61}$$

In this case, we can write

$$p(y \mid \boldsymbol{x}, \boldsymbol{w}, \sigma_n^2) = \mathcal{N}\left(f(\boldsymbol{x}, \boldsymbol{w}), \sigma_n^2\right) \tag{2.62}$$

Now, we want to use the training dataset $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y})$. where $\boldsymbol{X}$ is a $N \times M$ matrix and $\boldsymbol{y}$ is a $N \times 1$ vector, to determine the values of the unknown parameters $\boldsymbol{w}$ and σ_n^2. Making the assumption that the data points are drawn independently (2.16) from the distribution (2.62), we obtain the following expression

$$p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \sigma_n^2) = \prod_{i=1}^{N} \mathcal{N}\left(y_i \mid \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i), \sigma_n^2\right) \tag{2.63}$$

Taking the logarithm of (2.63) and making use of the standard form (2.34) for the univariate Gaussian we have the following likelihood

$$\log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \sigma_n^2) = -\frac{1}{2\sigma_n^2}\sum_{i=1}^{N}\left(y_i - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)^2 - \frac{N}{2}\log(\sigma_n^2) - \frac{N}{2}\log(2\pi) \tag{2.64}$$

Then, we can determine $\boldsymbol{w}$ and σ_n^2 as those that maximize (2.64). This is known as maximum likelihood estimation (MLE). Consider first the maximization with respect to $\boldsymbol{w}$. For this purpose, we can omit the last two terms on the right-hand side of (2.64) because they do not depend on $\boldsymbol{w}$. Also, we note that scaling the log-likelihood by a positive constant coefficient does not alter the location of the maximum with respect to $\boldsymbol{w}$, and so we can replace the coefficient $1/2\sigma_n^2$ with $1/2$.

Therefore, we have that instead of maximizing the log-likelihood, we can equivalently minimize the negative log-likelihood

$$\boldsymbol{w}^* = \arg\max_{w} \log(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \sigma_n^2) = \arg\min_{w} \frac{1}{2} \sum_{i=1}^{N} \left(y_i - \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i)\right)^2 \tag{2.65}$$

We can see that maximizing likelihood is equivalent, so far as determining $\boldsymbol{w}$ is concerned, to minimizing the sum-of-squares error function (2.3) that appeared in the polynomial curve fitting problem in Section 2.1. The sum-of-squares error function has arisen as a consequence of maximizing likelihood under the assumption of a Gaussian noise distribution. The solution of this problem is, therefore, (2.6)

$$\boldsymbol{w}^* = \left(\boldsymbol{\Phi}^T \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T \boldsymbol{y} \tag{2.66}$$

This is known as the normal equations for the least-squares problem. Here, $\boldsymbol{\Phi}$ is a $N \times M$ matrix, called the design matrix, which is written as

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi_0(\boldsymbol{x}_1) & \phi_1(\boldsymbol{x}_1) & \dots & \phi_{M-1}(\boldsymbol{x}_1) \\ \phi_0(\boldsymbol{x}_2) & \phi_1(\boldsymbol{x}_2) & \dots & \phi_{M-1}(\boldsymbol{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\boldsymbol{x}_N) & \phi_1(\boldsymbol{x}_N) & \dots & \phi_{M-1}(\boldsymbol{x}_N) \end{bmatrix} \tag{2.67}$$

Analogously to (2.9), we can also consider a regularization parameter yielding

$$\boldsymbol{w}^* = \left(\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{\Phi}^T \boldsymbol{y} \tag{2.68}$$

Additionally, we can maximize the log likelihood function (2.64) with respect to the noise variance σ_n^2, giving

$$\sigma_n^2 = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \boldsymbol{w}^{*T} \boldsymbol{\phi}(\boldsymbol{x}_i)\right)^2 \tag{2.69}$$

and so we see that the noise variance is given by the variance of the residuals of the target values y_i around the regression function.

In Figure 2.10, we show an example of a linear regression considering a set of radial basis functions (2.60) and the noise parameter σ_n^2. It is interesting to note that the parametrization and the number of basis functions considered play a key role in the resulting regression model.

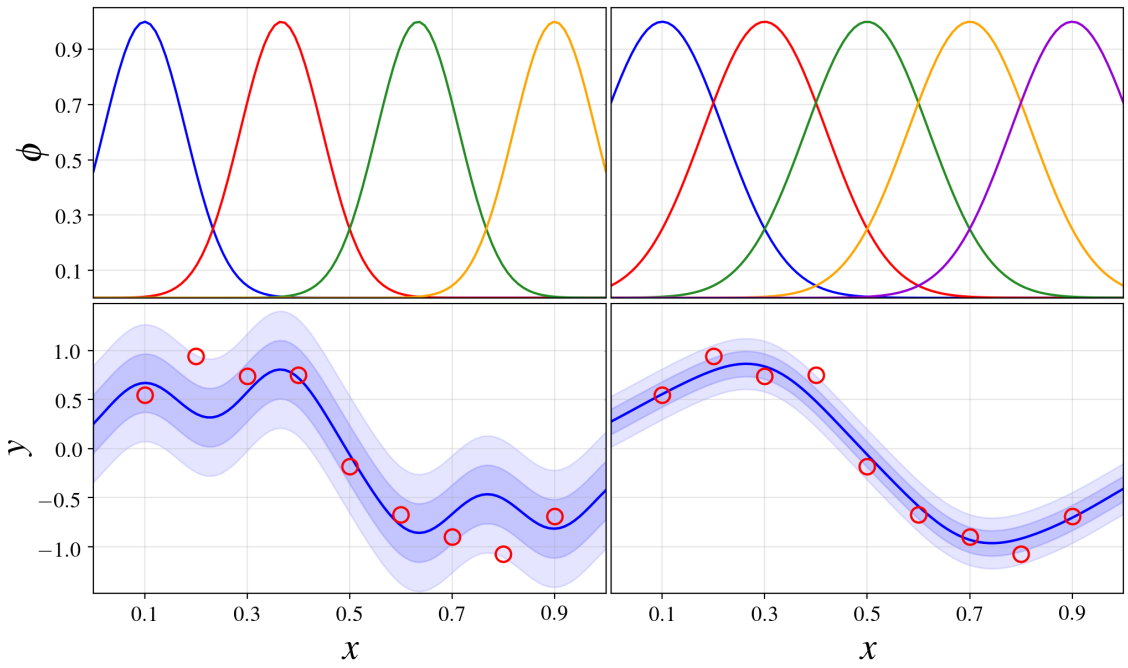

Figure 2.10: On the first row, the sets of radial basis functions (2.60) for the corresponding regression shown below. The red dots represent the training data, the solid blue line the mean of the predictive distribution (2.62), and the blue shaded areas one and two times the standard deviation of such distribution around the mean.

2.4.2 Bias–Variance Decomposition

Selecting the appropriate number of basis functions is key for avoiding overfitting but still allows enough flexibility of the model to capture the relevant trends in the data. Although the introduction of regularization terms can control overfitting for models with many parameters, this raises the question of how to determine a suitable value for the regularization coefficient λ.

We can formalize these issues through the introduction of a loss function $L(y, f(\boldsymbol{x}))$, also called a cost function, which is a single, overall measure of the performance of our regression model. Our goal, in probabilistic terms, is to minimize the total expected loss incurred, which is given by

$$\mathbb{E}\left[L\right] = \iint L(y, f(\boldsymbol{x}))p(y, \boldsymbol{x})d\boldsymbol{x}dy \tag{2.70}$$

A common choice of the loss function in regression problems is the squared loss

$$L(y, f(\boldsymbol{x})) = (f(\boldsymbol{x}) - y)^2 \tag{2.71}$$

Our goal is to choose $f(\boldsymbol{x})$ so as to minimize $\mathbb{E}[L]$. We can do this using the calculus of variations, obtaining

$$\frac{\delta \mathbb{E}[L]}{\delta f(\boldsymbol{x})} = 2 \int (f(\boldsymbol{x}) - y)\, p(y, \boldsymbol{x}) dy = 0 \tag{2.72}$$

Solving for $f(\boldsymbol{x})$, and using the sum (2.22) and product (2.23) rules of probability, we obtain

$$f(\boldsymbol{x}) = \frac{\int y p(y, \boldsymbol{x}) dy}{p(\boldsymbol{x})} = \int y p(y \mid \boldsymbol{x}) dy = \mathbb{E}[y \mid \boldsymbol{x}] \tag{2.73}$$

We can also derive this result in a slightly different way, which sheds light on the nature of the regression problem. Knowing that the optimal solution is the conditional expectation $\mathbb{E}[y \mid \boldsymbol{x}]$ (2.73), we can expand the loss function as follows

$$\begin{aligned} &(f(\boldsymbol{x}) - y)^2 = (f(\boldsymbol{x}) - \mathbb{E}[y \mid \boldsymbol{x}] + \mathbb{E}[y \mid \boldsymbol{x}] - y)^2 = \\ &(f(\boldsymbol{x}) - \mathbb{E}[y \mid \boldsymbol{x}])^2 + 2\,(f(\boldsymbol{x}) - \mathbb{E}[y \mid \boldsymbol{x}])\,(\mathbb{E}[y \mid \boldsymbol{x}] - y) + (\mathbb{E}[y \mid \boldsymbol{x}] - y)^2 \end{aligned} \tag{2.74}$$

Substituting into the loss function (2.71) and integrating, we see that the cross-term vanishes, resulting in an expression for the loss function in the form

$$\mathbb{E}[L] = \int (f(\boldsymbol{x}) - h(\boldsymbol{x}))^2 p(\boldsymbol{x}) d\boldsymbol{x} + \int (h(\boldsymbol{x}) - y)^2 p(\boldsymbol{x}) d\boldsymbol{x} \tag{2.75}$$

where we have substituted $h(\boldsymbol{x}) = \mathbb{E}[y \mid \boldsymbol{x}]$. The second term, which is independent of $f(\boldsymbol{x})$, arises from the intrinsic noise on the data and represents the minimum achievable value of the expected loss. The function $f(\boldsymbol{x})$ we seek to determine enters only in the first term, which is minimized when $f(\boldsymbol{x}) = h(\boldsymbol{x})$. However, in practice we have a dataset $\mathcal{D}$ containing only a finite number N of data points, and consequently we do not know $h(\boldsymbol{x})$ exactly. Consider the integrand of the first term in (2.75), which for a particular data set $\mathcal{D}$ takes the form

$$(f(\boldsymbol{x}, \mathcal{D}) - h(\boldsymbol{x}))^2 \tag{2.76}$$

If we add and subtract the expectancy of $f(\boldsymbol{x}, \mathcal{D})$ with respect to $\mathcal{D}$ in (2.76) and expand it, we obtain a result analogous to the one obtained in (2.74)

$$\begin{aligned} (f(\boldsymbol{x}, \mathcal{D}) - h(\boldsymbol{x}))^2 = &(f(\boldsymbol{x}, \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x}, \mathcal{D})])^2 + (\mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x}, \mathcal{D})] - h(\boldsymbol{x}))^2 + \\ &+ 2\,(f(\boldsymbol{x}, \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x}, \mathcal{D})])\,(\mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x}, \mathcal{D})] - f(\boldsymbol{x}, \mathcal{D})) \end{aligned} \tag{2.77}$$

We now take the expectation of this expression with respect to $\mathcal{D}$ and note that the final term vanishes, giving

$$\begin{aligned}&\mathbb{E}_{\mathcal{D}}\left[\left(f\left(\boldsymbol{x}, \mathcal{D}\right) - h(\boldsymbol{x})\right)^2\right] = \\ &\mathbb{E}_{\mathcal{D}}\left[\left(f(\boldsymbol{x}, \mathcal{D}) - \mathbb{E}_{\mathcal{D}}\left[f(\boldsymbol{x}, \mathcal{D})\right]\right)^2\right] + \left(\mathbb{E}_{\mathcal{D}}\left[f(\boldsymbol{x}, \mathcal{D})\right] - h(\boldsymbol{x})\right)^2\end{aligned} \tag{2.78}$$

We see that the expected squared difference between $f(\boldsymbol{x}, \mathcal{D})$ and the regression function $h(\boldsymbol{x})$ can be expressed as the sum of two terms. The first term in (2.78) is called the variance. It measures the extent to which the solutions for individual datasets vary around their average, and hence this measures the extent to which the function $f(\boldsymbol{x}, \mathcal{D})$ is sensitive to the particular choice of dataset. The second term, called the squared bias, represents the extent to which the average prediction over all datasets differs from the desired regression function. In other words, we obtain the following decomposition of the expected squared loss

$$\text{expected loss} \;=\; (\text{bias})^2 + \text{variance} + \text{noise} \tag{2.79}$$

This raises a central problem in supervised learning known as the bias-variance trade-off. Ideally, one wants to choose a model that both accurately captures the regularities in its training data, but also generalizes well to unseen data. Unfortunately, it is impossible to do both simultaneously. High-variance learning methods may be able to represent their training set well but are at risk of overfitting to noise. In contrast, algorithms with high bias produce simpler models that may fail to capture important regularities (i.e. underfit) in the data.

Now, we provide some intuition to support these definitions through an example. We generate a total of 20 training datasets, each containing $N = 9$ data points, from the following observational model

$$y = \sin(2\pi x) + \mathcal{N}\left(0, 0.01\right) \tag{2.80}$$

We then fit a model with 20 Gaussian basis functions (2.60) by minimizing the regularized error using (2.68). The obtained results are shown in Figure 2.11. The top row corresponds to a large value of the regularization coefficient λ that gives low variance, but high bias. Conversely, on the bottom row, for which λ is small, there is a large variance but low bias.

Although the bias-variance decomposition may provide some interesting insights into the model complexity problem from a frequentist perspective, it is of limited practical value, because the bias-variance decomposition is based on averages with respect to several datasets, whereas in practice we have only the single observed dataset.

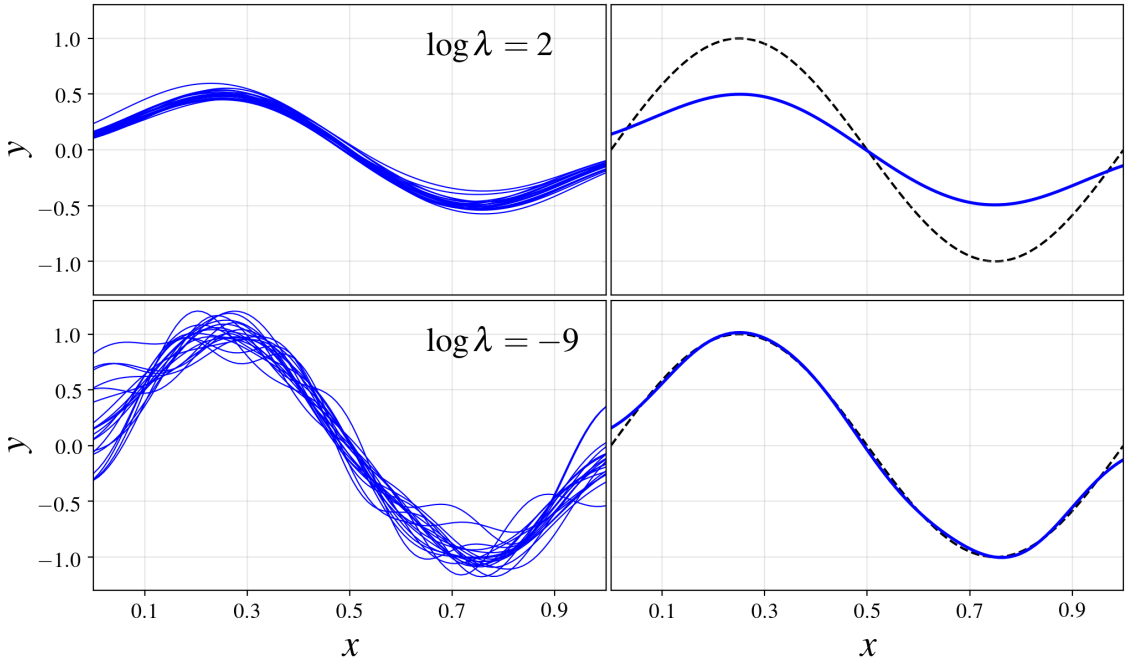

Figure 2.11: Dependence of bias and variance on the model complexity, governed by a regularization parameter λ. There are 20 data sets, each having $N = 9$ data points, and there are 20 Gaussian basis functions in the model. The left column shows the result of fitting the model to each dataset for two different values of λ. The right column shows the corresponding average of the 20 fits (blue solid line) along with the function (2.80) from which the datasets are generated (black dashed line).

2.4.3 Parameter Distribution

We now turn to a Bayesian treatment of linear regression, which avoids the over-fitting problem, and which also leads to automatic methods of determining model complexity using the training data alone. Since we are adopting a Bayesian approach, we need to introduce a prior probability distribution over the model parameters $\boldsymbol{w}$. For simplicity in the development, let such distribution be a zero-mean isotropic Gaussian of the form

$$p(\boldsymbol{w}) = \mathcal{N}\left(\mathbf{0}, \sigma_w^2 \boldsymbol{I}\right) \tag{2.81}$$

Using Bayes' theorem (2.31), the posterior distribution for $\boldsymbol{w}$, given the training dataset $\mathcal{D}$, is proportional to the product of the prior distribution and the likelihood function

$$p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X}) \propto p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}) p(\boldsymbol{w}) = \mathcal{N}\left(\boldsymbol{\Phi w}, \sigma_n^2 \boldsymbol{I}\right) \mathcal{N}\left(\mathbf{0}, \sigma_w^2 \boldsymbol{I}\right) \tag{2.82}$$

For deriving the posterior mean and variance of the resulting posterior we can proceed as in Section 2.3.1, focusing on the quadratic form in the exponent

$$\begin{aligned} &-\frac{1}{2\sigma_n^2}\left(\boldsymbol{y}-\boldsymbol{\Phi}\boldsymbol{w}\right)^T\left(\boldsymbol{y}-\boldsymbol{\Phi}\boldsymbol{w}\right)+\frac{1}{2\sigma_w^2}\boldsymbol{w}^T\boldsymbol{w}= \\ &-\frac{1}{2}\boldsymbol{w}^T\left(\frac{1}{\sigma_n^2}\boldsymbol{\Phi}^T\boldsymbol{\Phi}+\frac{1}{\sigma_w^2}\boldsymbol{I}\right)\boldsymbol{w}+\boldsymbol{w}^T\frac{1}{\sigma_n^2}\boldsymbol{\Phi}^T\boldsymbol{y}+\text{const} \end{aligned} \tag{2.83}$$

Comparing (2.40) and (2.83) we have that the posterior distribution is again Gaussian with mean and variance given by

$$p(\boldsymbol{w}\mid\boldsymbol{y},\boldsymbol{X})\propto\mathcal{N}\left(\boldsymbol{\mu}_{w|\mathcal{D}},\boldsymbol{\Sigma}_{w|\mathcal{D}}\right) \tag{2.84}$$

$$\boldsymbol{\Sigma}_{w|\mathcal{D}}=\left(\frac{1}{\sigma_n^2}\boldsymbol{\Phi}^T\boldsymbol{\Phi}+\frac{1}{\sigma_w^2}\boldsymbol{I}\right)^{-1} \tag{2.85}$$

$$\boldsymbol{\mu}_{w|\mathcal{D}}=\frac{1}{\sigma_n^2}\boldsymbol{\Sigma}_{w|\mathcal{D}}\boldsymbol{\Phi}^T\boldsymbol{y} \tag{2.86}$$

Note that if we consider an infinitely broad prior over the parameters $\sigma_w^2\longrightarrow\infty$ the mean of the posterior distribution reduces to (2.66). Similarly, if we have no data points $\sigma_n^2\longrightarrow\infty$, then the posterior distribution reverts to the prior (2.81). Also, we can see that the log of the posterior distribution is given by the sum of the log-likelihood and the log of the prior and, as a function of $\boldsymbol{w}$, takes the form

$$\log p(\boldsymbol{w}\mid\boldsymbol{y},\boldsymbol{X})=-\frac{1}{2\sigma_n^2}\sum_{i=1}^{N}\left(y_i-\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)^2-\frac{1}{2\sigma_w^2}\boldsymbol{w}^T\boldsymbol{w} \tag{2.87}$$

Thus, the maximization of (2.87) with respect to $\boldsymbol{w}$ is equivalent to the minimization of the sum-of-squares error function with the addition of a quadratic regularization term, being the solution equivalent to (2.68) taking $\lambda=\sigma_n^2/\sigma_w^2$.

An example of the computation of the posterior parameter distribution is shown in Figure 2.12. We consider a linear model of the form $f(x,\boldsymbol{w})=w_0+w_1x$. Note that once the training dataset is given, the posterior distribution is much more compact than the prior, and we see that sample functions drawn from the parameter distribution pass close to the data points. In the limit of an infinite number of data points, the posterior distribution would become a delta function centred on the true parameter values, shown as a white cross.

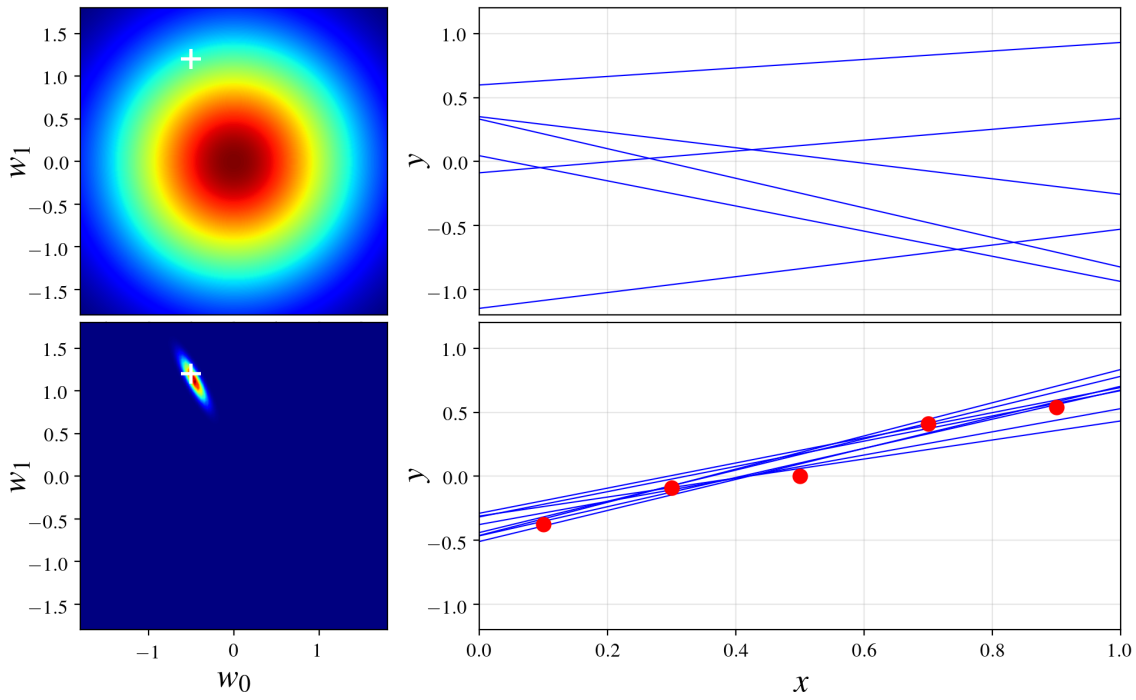

Figure 2.12: Computation of the parameter's $\boldsymbol{w}$ posterior distribution (2.84) for a linear model of the form $f(x, \boldsymbol{w}) = w_0 + w_1 x$. On the first row, the prior distribution after observing any data. Samples drawn from such distribution are shown on the right column. On the second row, the posterior distribution given the data represented as red dots. The true value of the parameters is depicted as a white cross.

2.4.4 Predictive Distribution

In practice, we are not interested in the value of $\boldsymbol{w}$ itself but rather in making predictions of y for new values of $\boldsymbol{x}$. This requires the computation of the predictive distribution defined by

$$p(y \mid \boldsymbol{x}, \mathcal{D}) = \int p(y \mid \boldsymbol{x}, \boldsymbol{w}) p(\boldsymbol{w} \mid \mathcal{D}) d\boldsymbol{w} \tag{2.88}$$

Since $p(y \mid \boldsymbol{x}, \boldsymbol{w})$ is given by (2.62), and the posterior parameter distribution is given by (2.84), we see that (2.88) involves the convolution of two Gaussians. The predictive distribution takes the form

$$p(y \mid \boldsymbol{x}, \mathcal{D}) = \mathcal{N}\left(\boldsymbol{\mu}_{w|\mathcal{D}}^T \boldsymbol{\phi}(\boldsymbol{x}), \sigma_n^2 + \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\Sigma}_{w|\mathcal{D}} \boldsymbol{\phi}(\boldsymbol{x})\right) \tag{2.89}$$

The first term in the variance of the predictive distribution (2.89) represents the noise on the data whereas the second term reflects the uncertainty associated with the parameters $\boldsymbol{w}$.

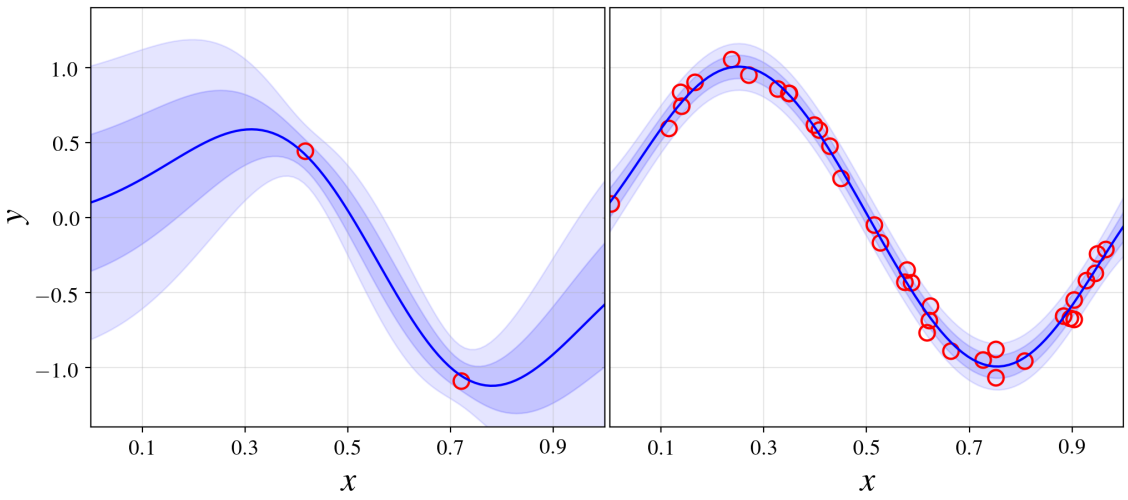

Figure 2.13: Predictive distribution (2.89) for a model consisting of 20 Gaussian basis functions given the data points depicted as red dots.

In Figure 2.13, we fit a model comprising a linear combination of Gaussian basis functions to datasets of two different sizes and then look at the corresponding predictive distributions. Note that the predictive uncertainty depends on x and is the smallest in the neighborhood of the data points. Also, we can see that the level of uncertainty decreases as more data points are observed.

2.4.5 Equivalent Kernel

The posterior mean solution in (2.89) for the linear basis function model has an interesting interpretation that sets the stage for kernel methods. The predictive mean can be rewritten in the form

$$f(\boldsymbol{x}, \boldsymbol{\mu}_{w|\mathcal{D}}) = \boldsymbol{\mu}_{w|\mathcal{D}}^T \boldsymbol{\phi}(\boldsymbol{x}) = \frac{1}{\sigma_n^2} \sum_{i=1}^{N} \boldsymbol{\phi}(\boldsymbol{x})^T \left(\frac{1}{\sigma_n^2} \boldsymbol{\Phi}^T \boldsymbol{\Phi} + \frac{1}{\sigma_w^2} \boldsymbol{I} \right)^{-1} \boldsymbol{\phi}(\boldsymbol{x}_i) y_i \tag{2.90}$$

Thus, the mean of the predictive distribution at a point $\boldsymbol{x}$ is given by a linear combination of the training set target variables y_i, so that we can write

$$f(\boldsymbol{x}, \boldsymbol{\mu}_{w|\mathcal{D}}) = \sum_{i=1}^{N} k(\boldsymbol{x}, \boldsymbol{x}_i) y_i \tag{2.91}$$

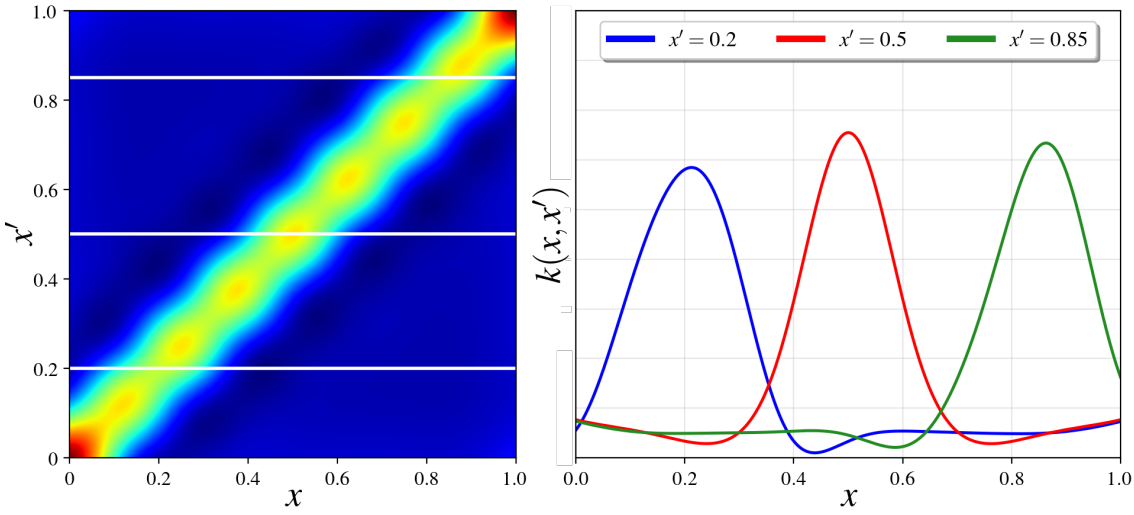

Figure 2.14: On the left, the equivalent kernel $k(x, x')$ for a set of Gaussian basis functions. On the right, three slices for different values of x', shown as white lines on the left plot.

The function $k(\boldsymbol{x}, \boldsymbol{x}')$ is known as the equivalent kernel. Note that it depends on the input values $\boldsymbol{x}_i$ from the training dataset. The equivalent kernel is illustrated for the case of Gaussian basis functions in Figure 2.14 in which the kernel functions $k(x, x')$ have been plotted as a function of x' for three different values of x. We see that they are localized around x, and so the mean of the predictive distribution at x, given by (2.90), is obtained by forming a weighted combination of the target values in which data points close to x are given higher weight than points further from x. Intuitively, it seems reasonable that we should weigh local observations more strongly than distant ones.

Further insight into the role of the equivalent kernel can be obtained by considering the covariance between $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$, which is given by

$$\text{cov}\left[f(\boldsymbol{x}), f(\boldsymbol{x}')\right] = \text{cov}\left[\boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{w}, \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}')\right] = \sigma_n^2 k(\boldsymbol{x}, \boldsymbol{x}') \tag{2.92}$$

From the form of the equivalent kernel, we can see that the predictive mean at nearby points is highly correlated, whereas for more distant pairs of points the correlation is smaller.

The formulation of linear regression in terms of a kernel function suggests an alternative approach to regression as follows. Instead of introducing a set of basis functions, which implicitly determines an equivalent kernel, we can instead define a localized kernel directly and use this to make predictions for new input vectors x, given the observed training set. This leads to a practical Bayesian framework for regression known as Gaussian Processes, which we discuss in detail in Chapter 3.

Chapter 3

Gaussian Processes

Gaussian Processes (GP) are concerned with supervised learning, which is the problem of learning input-output mappings from empirical data (the training dataset). This problem is also known as regression for continuous outputs. The main idea is that we need to move from the finite training data to a function that makes predictions for all possible input values.

Intuitively, a Gaussian Process can be seen as a distribution over functions. First, making some assumptions about the characteristic underlying function, a Gaussian Process defines a prior distribution over the functions that could explain our data. Higher probabilities are given to functions that we consider to be more likely, for example, because they are smoother. The great thing about Gaussian Processes is that the set of possible functions can be uncountably infinite. Second, conditional on the input-output data we have observed, a Gaussian Process then finds a posterior distribution of functions that explain the data.

In this chapter, we examine the main concepts behind Gaussian Processes. It serves as a first introduction to the field. For a more in-depth development, we refer the interested reader to [5], which is considered the canonical book on Gaussian Processes in the machine learning community. Also, [6] provides a very intuitive overview. The content of this chapter is mainly based on these two references.

In Section 3.1, we start by providing an intuitive introduction to Gaussian Processes in order to grasp the main concepts. Starting from fundamental ideas we suddenly wind up with the Gaussian Process regression equations. Next, in Section 3.2, we look into the formal definition and its interpretation. Afterward, in Section 3.3, we discuss the main component of Gaussian Process models: the covariance function. At the back end of this chapter, in Section 3.4, we address the fundamental problem of hyperparameter tuning and model selection.

3.1 An Intuitive Introduction to GP

In this section, we examine the basics behind Gaussian Process regression from a very intuitive point of view. We start in Section 3.1.1 by looking at a simple problem: approximating the value of variables that we can measure. These exact same concepts are then used in Section 3.1.2 to infer the value variables that we have not observed i.e. for regression to approximate functions. That is where we arrive at the Gaussian Process regression equations.

3.1.1 Approximating Measured Variables

Suppose that we have some variable f that we want to know its value. We expect it to be roughly μ, but it is likely to be anywhere in the interval $[\mu - \sigma_f, \mu + \sigma_f]$. Mathematically we treat f as a random variable. This random variable has a certain distribution. Based on our prior knowledge (knowledge that we have without doing any measurements), we can for instance say that f is a Gaussian random variable with mean μ and standard deviation σ_f

$$f \sim \mathcal{N}\left(\mu, \sigma_f^2\right) \tag{3.1}$$

where the $\sim$ sign means 'is distributed according to'. To learn more about the value of f, we do measurements. However, these are possibly affected by noise ε. As a result, we only get a measured value y, which is different from the true value f,

$$y = f + \varepsilon \tag{3.2}$$

We generally assume that we are dealing with Gaussian white noise. That is, ε is distributed according to a zero-mean Gaussian function with variance σ_n^2,

$$\varepsilon \sim \mathcal{N}\left(0, \sigma_n^2\right) \tag{3.3}$$

Its exact value is independent of the noise of any other measurement we might do and the value of f. As a result, prior to doing our measurement, we have

$$y \sim \mathcal{N}\left(\mu + 0, \sigma_f^2 + \sigma_n^2\right) \tag{3.4}$$

Next, when we perform a measurement, we get to know y deterministically. However, we do not know the measurement noise ε that was involved. As a result,

$$f = y - \varepsilon \sim \mathcal{N}\left(y, \sigma_n^2\right) \tag{3.5}$$

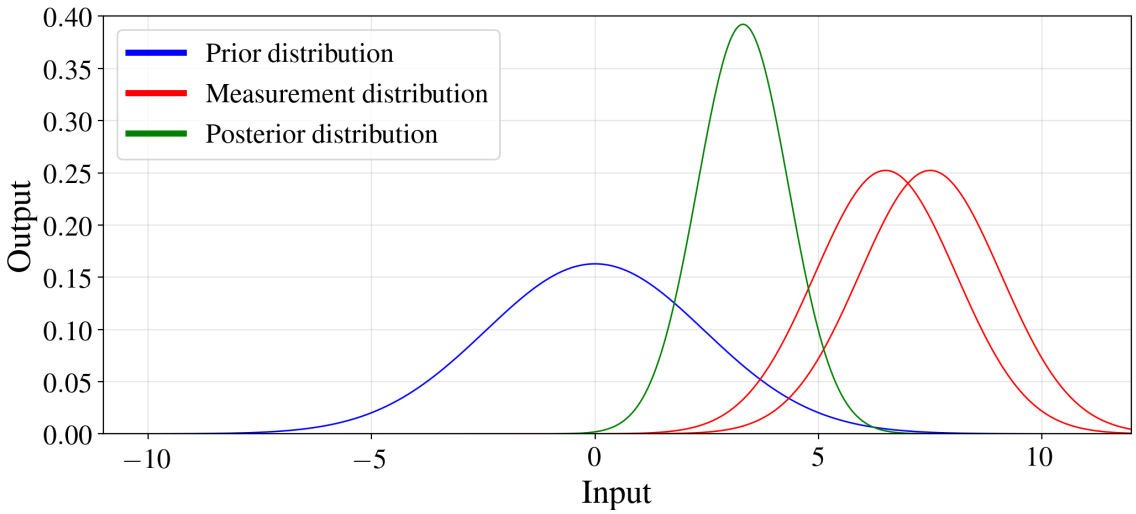

Figure 3.1: An example of computing the posterior distribution from the prior and the measurements of a single variable. Note that the resulting distribution is more peaked.

Note that distribution (3.1) differs from (3.5). How is this possible? Both have been obtained independently and both are correct. Actually, the first distribution corresponds to our prior assumption on the value of f. The second one, on the other hand, corresponds to the distribution given that we have measured y deterministically. What we can do, is combine both distributions to update our a priori assumption, and obtain an a posteriori distribution. Since (3.1) and (3.5) have been obtained independently, then the posterior distribution is given by

$$f \sim \frac{\mathcal{N}\left(\mu, \sigma_f^2\right) \mathcal{N}\left(y, \sigma_n^2\right)}{\int_{-\infty}^{\infty} \mathcal{N}\left(\mu, \sigma_f^2\right) \mathcal{N}\left(y, \sigma_n^2\right)} \tag{3.6}$$

Intuitively, the product in the numerator encodes the condition that f must be explained by (3.1) and (3.5). The denominator is a normalization constant. If we perform multiple measurements $y_1, y_2, \ldots$, we have instead

$$f \sim \frac{\mathcal{N}\left(\mu, \sigma_f^2\right) \mathcal{N}\left(y_1, \sigma_n^2\right) \mathcal{N}\left(y_2, \sigma_n^2\right) \ldots}{\int_{-\infty}^{\infty} \mathcal{N}\left(\mu, \sigma_f^2\right) \mathcal{N}\left(y_1, \sigma_n^2\right) \mathcal{N}\left(y_2, \sigma_n^2\right) \ldots} \tag{3.7}$$

An example of calculating this posterior distribution after two measurements can be seen in Figure 3.1. Note that the resulting posterior is more peaked than the prior's and the measurements'. This is reasonable, since as we obtain more measures we can expect a smaller uncertainty on the estimation i.e. a more narrow distribution.

Now, suppose that we have multiple variables f_1, f_2 and f_3 that we want to estimate. We can put them all into a vector $\boldsymbol{f}$, where the boldface indicates that $\boldsymbol{f}$ is a vector.

The prior distribution is now written as

$$\boldsymbol{f} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix}, \begin{bmatrix} \sigma_{f1}^2 & 0 & 0 \\ 0 & \sigma_{f2}^2 & 0 \\ 0 & 0 & \sigma_{f3}^2 \end{bmatrix} \right) = \mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}_f\right) \tag{3.8}$$

where $\boldsymbol{\mu}$ is the prior mean and $\boldsymbol{\Sigma}_f$ is the prior covariance matrix. Note that this is actually a multivariate Gaussian distribution. Similarly to (3.5), if we have measured values $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots$, variable $\boldsymbol{f}$ is distributed according to

$$\boldsymbol{f} \sim \mathcal{N}\left(\boldsymbol{y}, \boldsymbol{\Sigma}_n\right) \tag{3.9}$$

Now, as in (3.7), we have the posterior distribution

$$\boldsymbol{f} \sim \frac{\mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}_f\right)\mathcal{N}\left(\boldsymbol{y}_1, \boldsymbol{\Sigma}_n\right)\mathcal{N}\left(\boldsymbol{y}_2, \boldsymbol{\Sigma}_n\right)\ldots}{\int_{-\infty}^{\infty}\mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}_f\right)\mathcal{N}\left(\boldsymbol{y}_1, \boldsymbol{\Sigma}_n\right)\mathcal{N}\left(\boldsymbol{y}_2, \boldsymbol{\Sigma}_n\right)\ldots} \tag{3.10}$$

The great thing is that the resulting distribution will again be Gaussian,

$$\boldsymbol{f} \sim \mathcal{N}\left(\boldsymbol{\mu}', \boldsymbol{\Sigma}'\right) \tag{3.11}$$

$$\boldsymbol{\Sigma}' = \left(\boldsymbol{\Sigma}_f^{-1} + \boldsymbol{\Sigma}_n^{-1} + \boldsymbol{\Sigma}_n^{-1} + \ldots\right)^{-1} \tag{3.12}$$

$$\boldsymbol{\mu}' = \boldsymbol{\Sigma}'\left(\boldsymbol{\Sigma}_f^{-1}\boldsymbol{\mu} + \boldsymbol{\Sigma}_n^{-1}\boldsymbol{y_1} + \boldsymbol{\Sigma}_n^{-1}\boldsymbol{y_2} + \ldots\right) \tag{3.13}$$

It is interesting to know here that covariance matrices $\boldsymbol{\Sigma}_f$ and $\boldsymbol{\Sigma}_n$ are always positive definite. This means that the more measurements are added in (3.12), the smaller $\boldsymbol{\Sigma}'$ will get. And a smaller variance means a more accurate estimate. An example is shown in Figure 3.2. Combining the measurement distribution and the prior, the resulting posterior has smaller error bars.

3.1.2 Approximating Variables we Have not Measured

So far, we haven't really talked about making predictions of variables we have never performed any measurements on, which is actually the motivation behind supervised learning. Suppose that we have some function $f(x)$ and we want to estimate or approximate the value of this function at various input points x_1, x_2 and x_3. Well, we again have three numbers that we want to know, so just like in the previous subsection, we assume that they each have some prior mean $m(x)$ and variance $\sigma_f(x)^2$. Note that these may now also depend on the function input x.

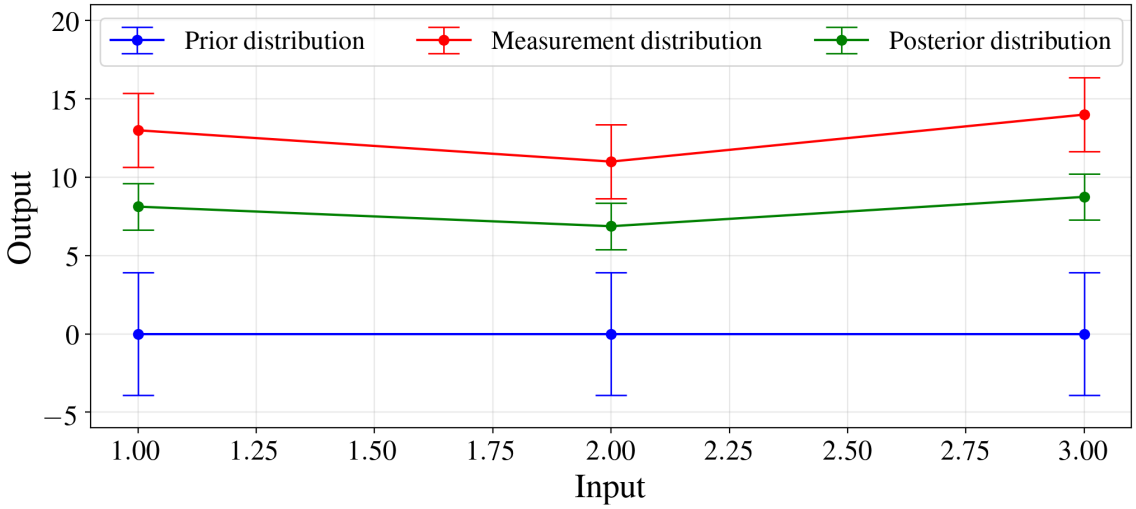

Figure 3.2: An example of computing the posterior distribution from the prior and the measurements of three variables. The error bars indicate the 95% confidence interval, which is equivalent to 1.96 times the standard deviation.

The problem now is that, if we now know something about $f(x_1)$, we still cannot say anything about $f(x_2)$ or $f(x_3)$ if these are not related somehow. Mathematically, this is because matrix $\mathbf{\Sigma}_f$ is diagonal. So we have to assume some kind of relationship between these function values. We can, for instance, assume that the original function $f(x)$ is smooth and doesn't vary too much with small variations of x. This connection can be expressed in probabilistic terms as a correlation.

Let's have a closer look into the aforementioned smoothness assumption. If x_1 is close to x_2, $f(x_1)$ should have a similar value than $f(x_2)$. On the other hand, if x_1 is further apart from x_3, we can say less about the value of $f(x_3)$ from the value of $f(x_1)$. From a probabilistic point of view, we can assume that $f(x_1)$ and $f(x_2)$ are strongly correlated, while $f(x_1)$ and $f(x_3)$ are weakly correlated. To express this mathematically, we define a correlation function $c(x, x')$ which defines the a priori relation between the function values $f(x)$ and $f(x')$ for any input points x and x'. For instance, we could use a Squared Exponential correlation function (SE correlation function)

$$c(x, x') = \exp\left(-\frac{1}{2}\frac{(x - x')^2}{\ell^2}\right) \tag{3.14}$$

Here ℓ is a length-scale for the input x. With this function, nearby points are strongly correlated, while the function values for input points that are far away from each other, relative to ℓ, have a nearly zero correlation. In Figure 3.3 the correlation function (3.14) is illustrated for different values of the parameter ℓ.

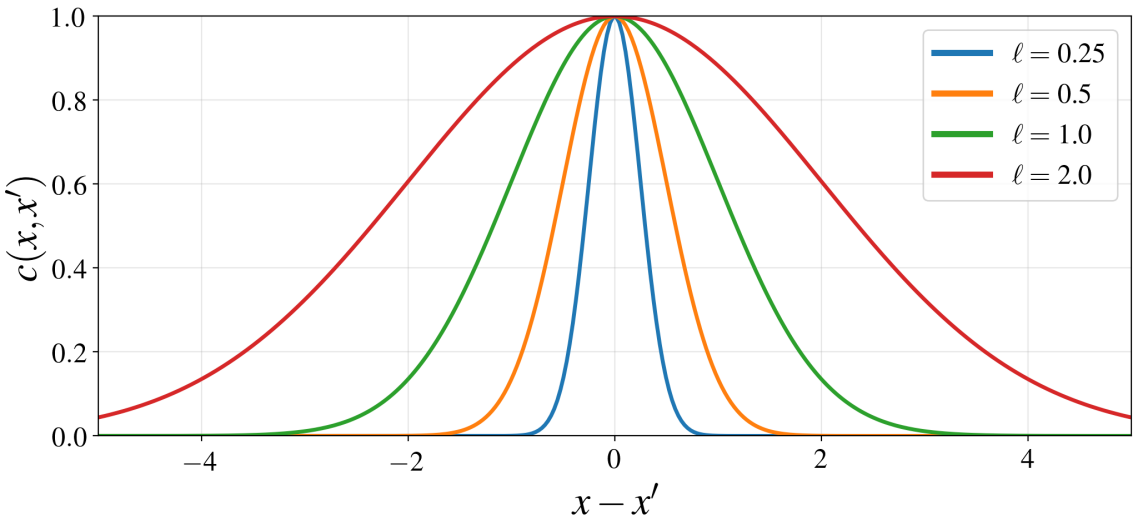

Figure 3.3: The correlation function (3.14) for different values of ℓ.

In practice, we actually don't use a correlation function but we use something very similar, the covariance function $k(x, x')$, which is related by

$$k(x, x') = \sigma_f(x)\sigma_f(x')c(x, x') \tag{3.15}$$

Assuming a constant standard deviation $\sigma_f(x) = \sigma_f$, we obtain the well-known Squared Exponential covariance function (SE covariance function)

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{1}{2}\frac{(x - x')^2}{\ell^2}\right) \tag{3.16}$$

Then, our prior distribution for a three-dimensional case is

$$\begin{aligned} \boldsymbol{f} = f(X) = \begin{bmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \end{bmatrix} \sim \mathcal{N}\left(m(X), k(X, X)\right) = \mathcal{N}\left(\boldsymbol{m}, \boldsymbol{K}\right) = \\ = \mathcal{N}\left(\begin{bmatrix} m(x_1) \\ m(x_2) \\ m(x_3) \end{bmatrix}, \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & k(x_1, x_3) \\ k(x_2, x_1) & k(x_2, x_2) & k(x_2, x_3) \\ k(x_3, x_1) & k(x_3, x_2) & k(x_3, x_3) \end{bmatrix}\right) \end{aligned} \tag{3.17}$$

Next, suppose that we have measured deterministically (without noise) that $f(x_1) = f_1$. What does this tell us about the distribution of $f(x_2)$ and $f(x_3)$? In probability theory, this question can be answered conditioning the distribution (3.17) on the measured value $f(x_1) = f_1$. For $f(x_2)$ we have

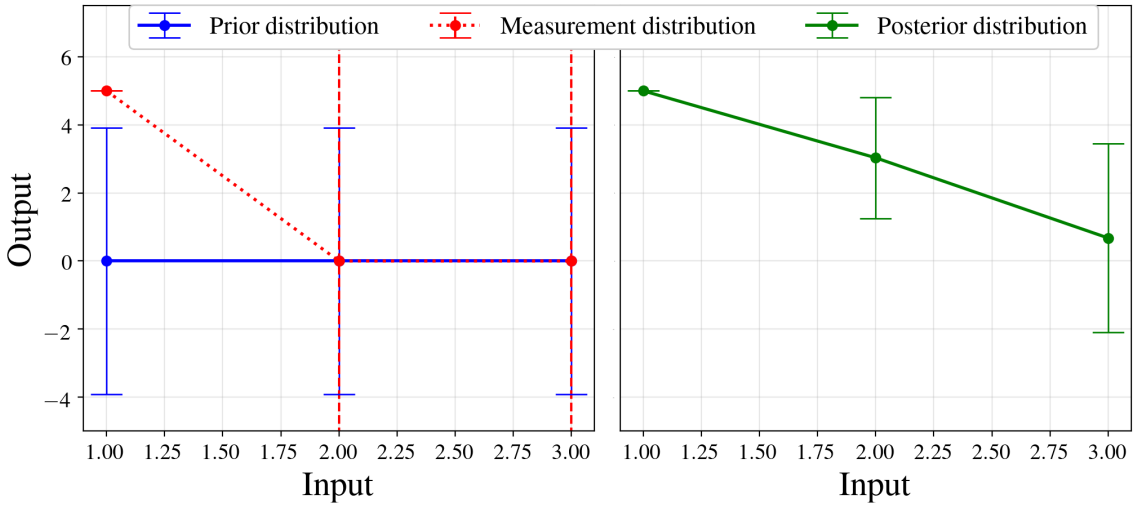

Figure 3.4: A first example of a Gaussian Process. First, we assume a prior distribution (blue). Then, we measure a value of the output only for input location 1 deterministically. The remaining are unknown (infinite variance). On the right we have the distribution conditional on the measurement, assuming a relation between outputs according to covariance function (3.16) with $\ell = 0.5$ and $\sigma_f^2 = 1.0$.

$$f(x_2) \mid f(x_1) = f_2 \sim \mathcal{N}(m(x_2) + k(x_2, x_1)k(x_1, x_1)^{-1}\,(f_1 - m(x_1)) \\ k(x_2, x_2) - k(x_2, x_1)k(x_1, x_1)^{-1}k(x_1, x_2)) \tag{3.18}$$

and analogously for $f(x_3)$. The conditional probability bar $\mid$ can be read as 'given that'. Note that this is equivalent to a prediction since we are estimating the value of function $f(x)$ in input locations where a measure is not available, given a known data point, which is what supervised learning is all about. An example is shown in Figure 3.4. Note that although we have only a measure on one input location, the uncertainty on the remaining has gotten significantly smaller, especially for the locations close to the measurement.

At this moment, we can translate these ideas into a more general mathematical formulation. We distinguish between measurement points x_i for $i = 1, \ldots, n$, with n the number of measurements, and points where we want to predict the function value x_{*j} for $j = 1, \ldots, n_*$, with n_* the number of prediction points. All these points form a measurement (training) input set X and a prediction input set X_*. Analogously to (3.17), the prior distribution of $\boldsymbol{f}$ and $\boldsymbol{f}_*$ is now given by

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{m} \\ \boldsymbol{m}_* \end{bmatrix}, \begin{bmatrix} \boldsymbol{K} & \boldsymbol{K}_* \\ \boldsymbol{K}_*^T & \boldsymbol{K}_{**} \end{bmatrix} \right) \tag{3.19}$$

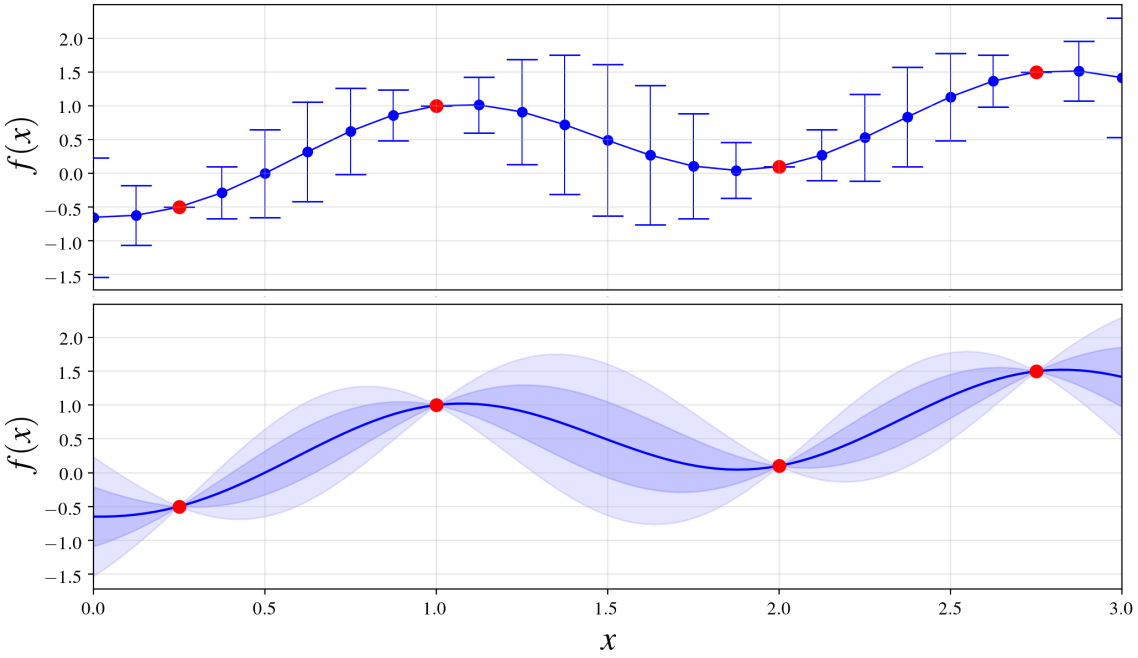

Figure 3.5: An example of a Gaussian Process. The red dots represent the measurements. For the figure above we have used $n_* = 25$ prediction points. Error bars represent two times the standard deviation. Below, we have used $n_* = 100$ prediction points. We have replaced the error bars with a colored area, where the inner (darker) area is one time the standard deviation and the outer (lighter) area is two times the standard deviation.

where $\boldsymbol{K}_* = K(X, X_*)$ denotes the $n \times n_*$ matrix of the covariances evaluated at all pairs of training and prediction points, and similarly for the other entries $k(X, X) = \boldsymbol{K}$ and $\boldsymbol{K}_{**} = k(X_*, X_*)$. Additionally, we write $\boldsymbol{f} = f(X)$, $\boldsymbol{m} = m(X)$ and identically for the prediction set.

Therefore, as (3.18), the posterior distribution of $\boldsymbol{f}_*$, given the measured function values $\boldsymbol{f}$, now equals

$$\boldsymbol{f}_* \sim \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \tag{3.20}$$

$$\boldsymbol{\mu}_* = \boldsymbol{m}_* + \boldsymbol{K}_*^T \boldsymbol{K}^{-1} (\boldsymbol{f} - \boldsymbol{m}) \tag{3.21}$$

$$\boldsymbol{\Sigma}_* = \boldsymbol{K}_{**} - \boldsymbol{K}_*^T \boldsymbol{K}^{-1} \boldsymbol{K}_* \tag{3.22}$$

The main advantage is that, with this expression, we can incorporate as many measurement points and as many prediction points as we want without modifying the formulation. An example is illustrated in Figure 3.5, where we predict $n_* = 25$ and $n_* = 100$ function values given four training points. The representation of the Gaussian Process in the bottom figure is the one adopted for the remainder of the chapter. Note that a Gaussian Process is characterized by both its mean and its covariance (i.e. uncertainty).

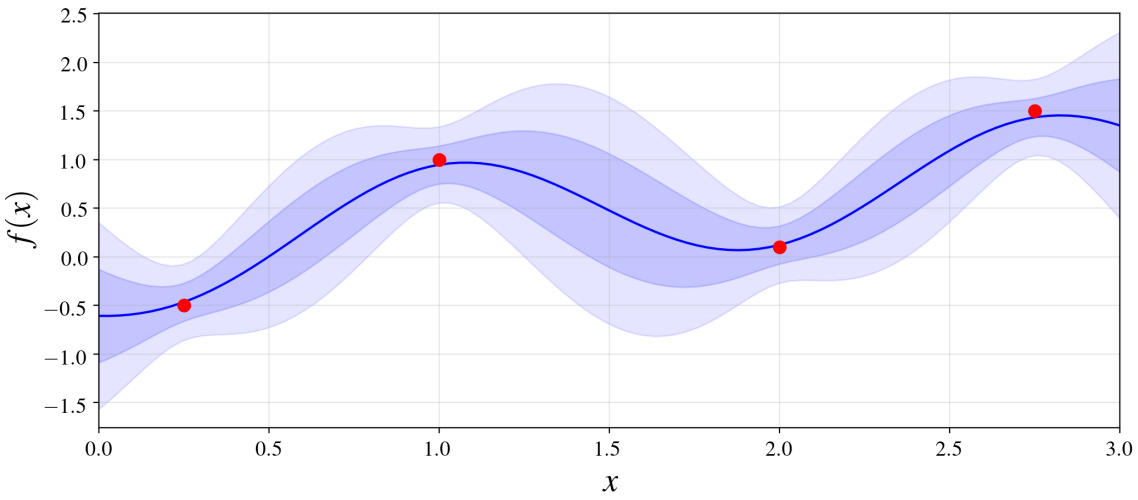

Figure 3.6: An updated version of Figure 3.5, in which measurement noise is considered.

Until now we have assumed that $\boldsymbol{f}$ is measured deterministically. But, what if this is not the case? If there is measurement noise (3.3), we have

$$\boldsymbol{y} = \boldsymbol{f} + \boldsymbol{\varepsilon} \tag{3.23}$$

For a multi-dimensional variable, we assume that the noise is distributed as

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_m \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_n^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n^2 \end{bmatrix} \right) = \mathcal{N}\left(\boldsymbol{0}, \sigma_n^2 \boldsymbol{I}\right) \tag{3.24}$$

where $\boldsymbol{I}$ refers to the $n \times n$ identity matrix. Thus, analogously to (3.4), the prior distribution of the noisy measurement vector $\boldsymbol{y}$ equals

$$\boldsymbol{y} \sim \mathcal{N}\left(\boldsymbol{m}, \boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right) \tag{3.25}$$

Considering noise, identically to (3.20), (3.21) and (3.22), the posterior yields

$$\boldsymbol{f}_* \sim \mathcal{N}\left(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*\right) \tag{3.26}$$

$$\boldsymbol{\mu}_* = \boldsymbol{m}_* + \boldsymbol{K}_*^T \left(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \left(\boldsymbol{y} - \boldsymbol{m}\right) \tag{3.27}$$

$$\boldsymbol{\Sigma}_* = \boldsymbol{K}_{**} - \boldsymbol{K}_*^T \left(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{K}_* \tag{3.28}$$

So, what is the effect of adding measurement noise to our predictions? Basically, the posterior uncertainties (variance) will be slightly bigger, which is reasonable if we measure with less precision. For instance, compare Figures 3.5 and 3.6.

3.2 The Formal Definition of GP Regression

Now, we know the main concepts behind Gaussian Process (GP) regression. But we haven't really looked at what a Gaussian Process actually is. It is time to look at the formal definition. Consider a training set $\mathcal{D}$ with n observations,

$$\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \mid i = 1, \ldots, n\} \tag{3.29}$$

where $\boldsymbol{x}$ denotes a input vector of dimension D, and y denotes a scalar output or target; the column vector inputs for all n cases are aggregated in the $D \times n$ design matrix X, and the targets are collected in the vector $\boldsymbol{y}$, so we can write $\mathcal{D} = (X, \boldsymbol{y})$. In the regression setting, the targets are real values. We are interested in making inferences about the relationship between inputs and targets, i.e. the conditional distribution of the targets given the inputs.

One can think of a Gaussian Process as defining a distribution over functions, and inference taking place directly in the space of functions. More formally, a Gaussian Process is a collection of random variables, any finite number of which have a joint Gaussian distribution. Let's take a look at exactly how this works.

Similarly to a multivariate Gaussian distribution $\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}_f)$, which can be defined just by the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}_f$; a Gaussian Process is completely specified by its mean function and covariance function. We define the mean function $m(\boldsymbol{x})$ and the covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ of a real process $f(\boldsymbol{x})$ as

$$m(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})] \tag{3.30}$$
$$k(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x}') - m(\boldsymbol{x}'))] \tag{3.31}$$

where $\mathbb{E}[]$ refers to the expectation. Then, we can write the Gaussian process as

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')) \tag{3.32}$$

The specification of the covariance function implies a distribution over functions. For instance, consider the Squared Exponential covariance function (3.16). To see this, we can draw samples from the distribution of functions evaluated at any number of points; in detail, we choose a number of input points, X_* and write out the corresponding covariance matrix element-wise. Then we generate a random Gaussian vector with this covariance matrix and mean function zero

$$\boldsymbol{f}_* \sim \mathcal{N}(\boldsymbol{0}, k(X_*, X_*)) \tag{3.33}$$

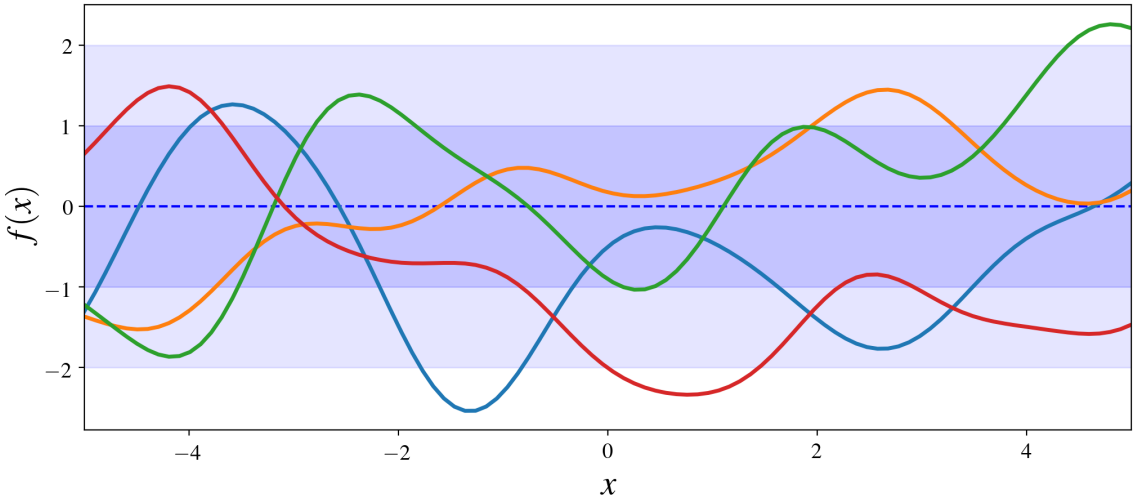

Figure 3.7: Four functions drawn at random from a GP prior with $m(\boldsymbol{x}) = 0$ and a SE covariance function (3.16).

and plot the generated values as a function of the inputs. Figure 3.7 shows four such samples. Each of the samples is a possible function from our prior distribution. Notice that the functions look smooth. In fact, the SE covariance function is infinitely differentiable, leading to the process being infinitely differentiable.

However, we are usually not primarily interested in drawing random functions from the prior but want to incorporate the knowledge that the training data provides about the function. Let us assume that we do not have access to function values themselves, but only noisy versions $y = f(\boldsymbol{x}) + \varepsilon$. Assuming additive independent identically distributed Gaussian noise ε with variance σ_n^2, the prior on the noisy observations becomes (3.25). We can write the joint distribution of the observed target values and the function values at the test locations X_* under the prior as

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(X) \\ m(X_*) \end{bmatrix}, \begin{bmatrix} k(X,X) + \sigma_n^2 \boldsymbol{I} & k(X,X_*) \\ k(X_*,X) & k(X_*,X_*) \end{bmatrix} \right) \tag{3.34}$$

To get the posterior distribution over functions we need to restrict this joint prior distribution to contain only those functions which agree with the observed data points. In probabilistic terms, this operation corresponds to condition the joint Gaussian prior distribution on the observations to give

$$\boldsymbol{f}_* \sim \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \tag{3.35}$$

$$\boldsymbol{\mu}_* = m(X_*) + k(X_*,X)\left(k(X,X) + \sigma_n^2 \boldsymbol{I}\right)^{-1}(\boldsymbol{y} - m(X)) \tag{3.36}$$

$$\boldsymbol{\Sigma}_* = k(X_*,X_*) - k(X_*,X)\left(k(X,X) + \sigma_n^2 \boldsymbol{I}\right)^{-1} k(X,X_*) \tag{3.37}$$

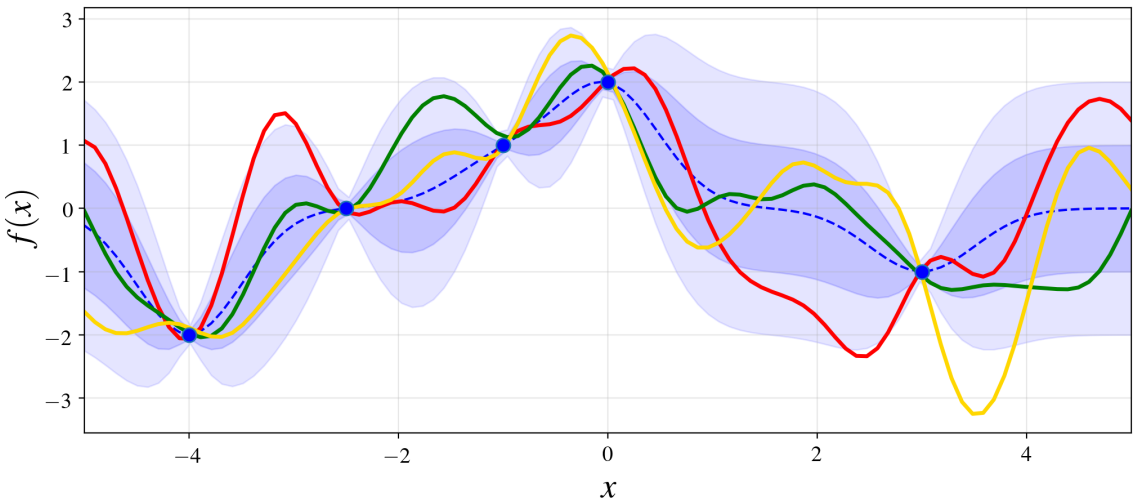

Figure 3.8: Three functions drawn at random from the posterior, i.e. the prior conditioned on the five observations indicated as blue dots.

which is analogous to (3.26), (3.27) and (3.28). Function values $\boldsymbol{f}_*$ (corresponding to X_*) can be sampled from the joint posterior distribution by evaluating the mean and covariance matrix from (3.36) and (3.37), and generating samples according to the method described previously for the prior distribution (Figure 3.7). Figure 3.8 shows the results of these computations given the five data points marked as blue dots. Note that all the sampled functions explain the observed data. In the case that there is only one test point, equations (3.36) and (3.37) are reduced to

$$\mathbb{E}\left[f(\boldsymbol{x}_*)\right] = m(\boldsymbol{x}_*) + k(\boldsymbol{x}_*, X)\left(k(X,X) + \sigma_n^2\boldsymbol{I}\right)^{-1}(\boldsymbol{y} - m(X)) \tag{3.38}$$

$$\mathbb{V}\left[f(\boldsymbol{x}_*)\right] = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k(\boldsymbol{x}_*, X)\left(k(X,X) + \sigma_n^2\boldsymbol{I}\right)^{-1}k(X, \boldsymbol{x}_*) \tag{3.39}$$

where $\mathbb{V}$ refers to the variance. Note first that the mean prediction (3.38) is a linear combination of observations $\boldsymbol{y}$. Another way to look at this equation is to see it as a linear combination of n kernel functions, each one centered on a training point,

$$\mathbb{E}\left[f(\boldsymbol{x}_*)\right] = m(\boldsymbol{x}_*) + \sum_{i=1}^{n}\alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}_*) \tag{3.40}$$

where $\boldsymbol{\alpha} = \left(k(X,X) + \sigma_n^2\boldsymbol{I}\right)^{-1}(\boldsymbol{y} - m(X))$. This vector does not depend on $\boldsymbol{x}_*$, so once all our measurements are known, we only have to compute it once. So what we see is that the posterior mean function is a sum of n covariance functions. Mathematically, we can also see this as a sum of basis functions. And we use just as many basis functions as we have measurements. The interesting thing is that other function approximation methods like neural networks and support vector machines, mathematically come down to the same.

Algorithm 1: Posterior predictive distribution for Gaussian Processes

Input: X (inputs), $\boldsymbol{y}$ (targets), $m()$ (mean function), $k(,)$ (covariance function), σ_n^2 (noise level), X_* (test inputs)

$\boldsymbol{K} = k(X, X), \quad \boldsymbol{K}_* = k(X, X_*), \quad \boldsymbol{K}_{**} = k(X_*, X_*);$

$\boldsymbol{L} = \text{cholesky}(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I});$

$\boldsymbol{K}_{inv} = \left(\boldsymbol{L}^{-1}\right)^T \boldsymbol{L}^{-1};$

$\boldsymbol{\mu}_* = m(X_*) + \boldsymbol{K}_*^T \boldsymbol{K}_{inv} \left(\boldsymbol{y} - m(X)\right);$

$\boldsymbol{\Sigma}_* = \boldsymbol{K}_{**} - \boldsymbol{K}_*^T \boldsymbol{K}_{inv} \boldsymbol{K}_*;$

return: $\boldsymbol{\mu}_*$ (posterior mean), $\boldsymbol{\Sigma}_*$ (posterior covariance)

Finally, a practical implementation of Gaussian Process regression is shown in Algorithm 1. The algorithm uses Cholesky decomposition, instead of directly inverting the matrix since it is faster and numerically more stable. The algorithm returns the posterior predictive mean and variance.

3.3 Covariance Functions

The covariance function is the key ingredient in a Gaussian Process predictor, as it encodes our assumptions about the function which we wish to learn. From a slightly different viewpoint, it is clear that in supervised learning the notion of similarity between data points is crucial. It is a basic assumption that points with inputs $\boldsymbol{x}$ which are close are likely to have similar target values y, and thus training points that are near to a test point should be informative about the prediction at that point. Under the Gaussian process view, it is the covariance function that defines nearness or similarity.

An arbitrary function of input pairs $\boldsymbol{x}$ and $\boldsymbol{x}'$ will not, in general, be a valid covariance function. A general name for a function $k(,)$ of two arguments mapping a pair of inputs $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$ into $\mathbb{R}$ is a kernel. Given a set of input points $\{\boldsymbol{x}_i \mid i = 1, \ldots, n\}$ we can compute the Gram matrix $\boldsymbol{K}$ whose entries are $\boldsymbol{K}_{ij} = k(x_i, x_j)$. If $k(,)$ is a covariance function we call the matrix $\boldsymbol{K}$ the covariance matrix. A real kernel is said to be symmetric if $k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}', \boldsymbol{x})$; clearly covariance functions must be symmetric from the definition. This implies that the Gram matrix corresponding to a covariance function must be positive semidefinite.

We start in Section 3.3.1 providing some examples of some commonly-used covariance functions and examining their properties. Afterward, in Section 3.3.2, we discuss some ways to build complex covariance functions from the combination of more simple ones.

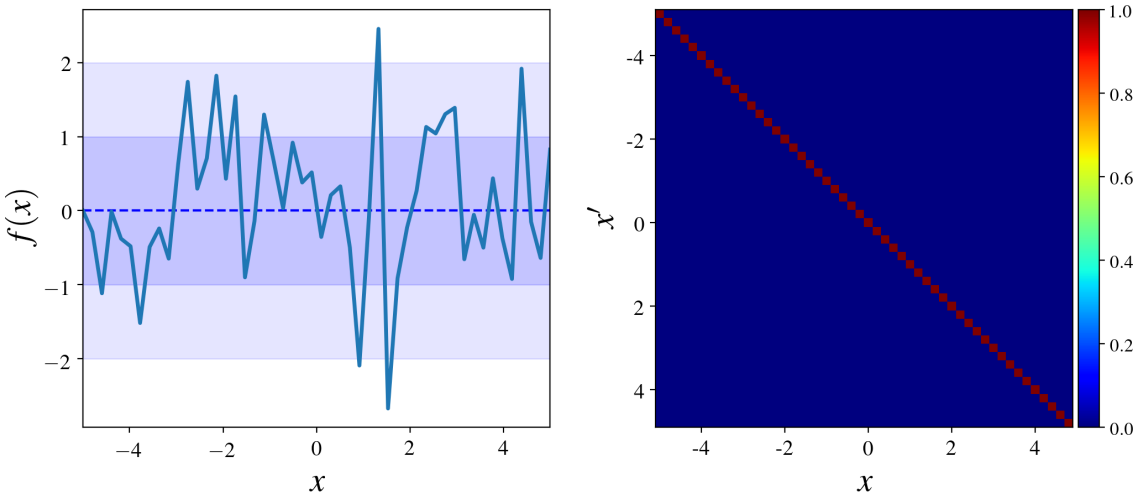

Figure 3.9: On the left, a random function drawn from a zero-mean Gaussian Process with white noise covariance function (3.41) for $\sigma_n^2 = 1$. On the right, a visual representation of the covariance matrix.

3.3.1 Examples of Covariance Functions

White Noise Covariance Function

The white noise kernel represents independent and identically distributed noise

$$k(x_i, x_j) = \sigma_n^2 \delta_{ij} \tag{3.41}$$

where δ_{ij} is a Kronecker delta which is one if $i = j$ and zero otherwise. The parameter σ_n^2 represents the noise variance. In Figure 3.9, we show a sample from this prior. We can see that the resulting function is similar to a noise signal. We can also see that the covariance matrix elements are zero everywhere except on the diagonal. This is because the noise is uncorrelated.

Squared Exponential Covariance Function

The squared exponential (SE) covariance function has already been introduced in (3.16) and has the form

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{1}{2}\frac{(x - x')^2}{\ell^2}\right) \tag{3.42}$$

with parameter ℓ defining the characteristic length-scale and σ_f^2 the output variance.

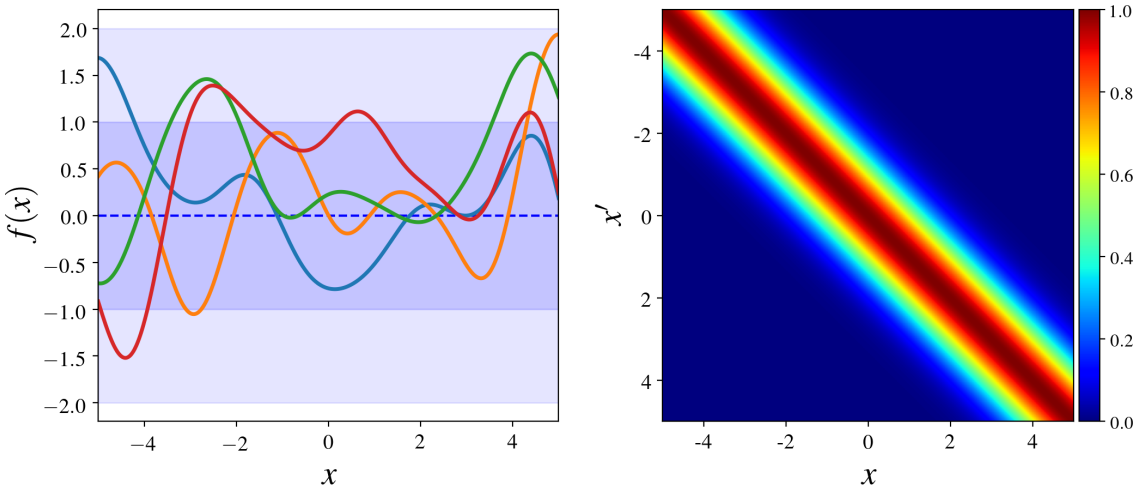

Figure 3.10: On the left, random functions drawn from a zero-mean Gaussian Process with SE covariance function (3.42), for $\sigma_f = 1$ and $\ell = 1$. On the right, a visual representation of the covariance matrix.

This type of covariance function is known as stationary since depends on $\boldsymbol{x} - \boldsymbol{x}'$. Thus it is invariant to translations in the input space. Additionally, since is a function only of $|\boldsymbol{x} - \boldsymbol{x}'|$ it is also called isotropic, which means that it is invariant to all rigid motions. Furthermore, it is infinitely differentiable, which means that the GP with this covariance function has derivatives of all orders, and is thus very smooth. Such strong smoothness assumptions are unrealistic for modeling many physical processes. However, the squared exponential is probably the most widely-used kernel within the kernel machines field. The form of the SE kernel matrix and samples drawn from it for $\sigma_f = 1$ and $\ell = 1$ are shown in Figure 3.10.

Matérn Covariance Function

The Matérn class of covariance functions is given by

$$k(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}|x - x'|}{\ell} \right)^{\nu} K_v \left(\frac{\sqrt{2\nu}|x - x'|}{\ell} \right) \tag{3.43}$$

with positive parameters ν and ℓ, where K_ν is a modified Bessel function. For $\nu \to \infty$, it converges to the SE covariance function (3.42). The most interesting case for machine learning is $\nu = 5/2$, for which

$$k(x, x') = \left(1 + \frac{\sqrt{5}|x - x'|}{\ell} + \frac{5(x - x')^2}{3\ell^2} \right) \exp \left(-\frac{\sqrt{5}|x - x'|}{\ell} \right) \tag{3.44}$$

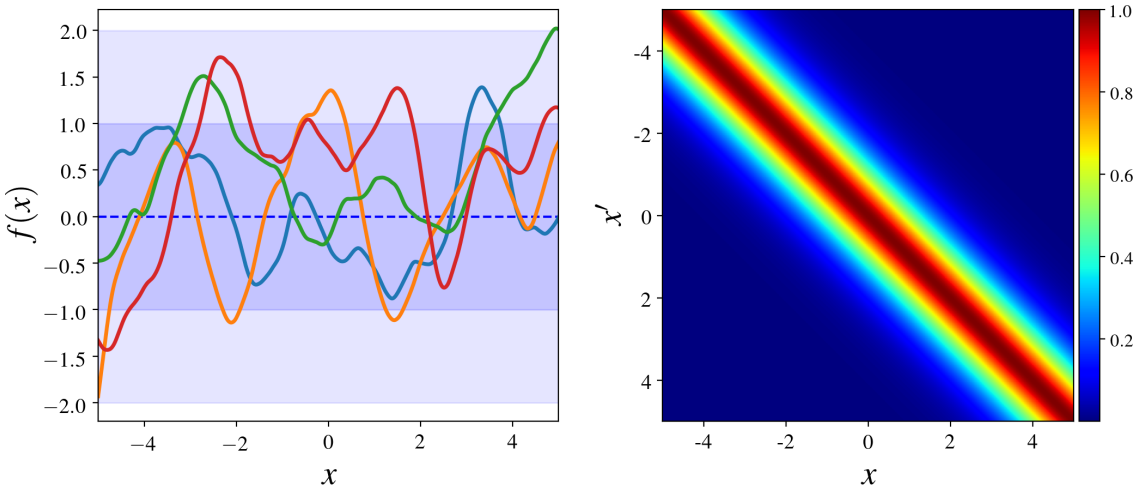

Figure 3.11: On the left, random functions drawn from a zero-mean Gaussian Process with Matérn covariance function (3.44), for $\ell = 1$. On the right, a visual representation of the covariance matrix.

This kernel is suitable for modeling processes that do not satisfy the strong smoothness assumption that imposes the SE covariance function. The form of the Matérn kernel matrix and samples drawn from it for $\ell = 1$ are shown in Figure 3.11. Note that the samples drawn in Figure 3.10 are much smoother.

Rational Quadratic Covariance Function

The rational quadratic (RQ) covariance function

$$k(x, x') = \left(1 + \frac{(x - x')^2}{2\alpha\ell^2}\right)^{-\alpha} \tag{3.45}$$

with $\alpha, \ell > 0$ can be seen as an infinite sum of SE covariance functions with different characteristic length scales. So, GP priors with this kernel expect to see functions that vary smoothly across many length scales. The parameter α determines the relative weighting of large-scale and small-scale variations. When $\alpha \to \infty$, the RQ kernel is identical to the SE kernel. This is in fact the most general representation for an isotropic kernel. Figure 3.12 illustrates the behaviour for $\ell = 1$ and $\alpha = 1$.

Periodic Covariance Function

The periodic kernel allows us to model periodic functions

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{2}{\ell^2}\sin^2\left(\frac{\pi|x - x'|}{p}\right)\right) \tag{3.46}$$

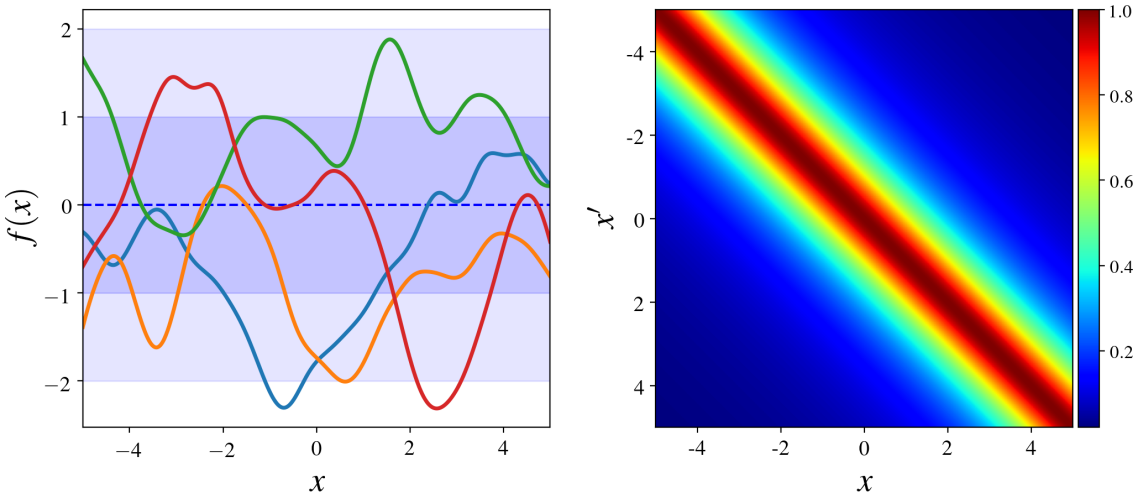

Figure 3.12: On the left, random functions drawn from a zero-mean Gaussian Process with rational quadratic covariance function (3.45), for $\ell = 1$ and $\alpha = 1$. On the right, a visual representation of the covariance matrix.

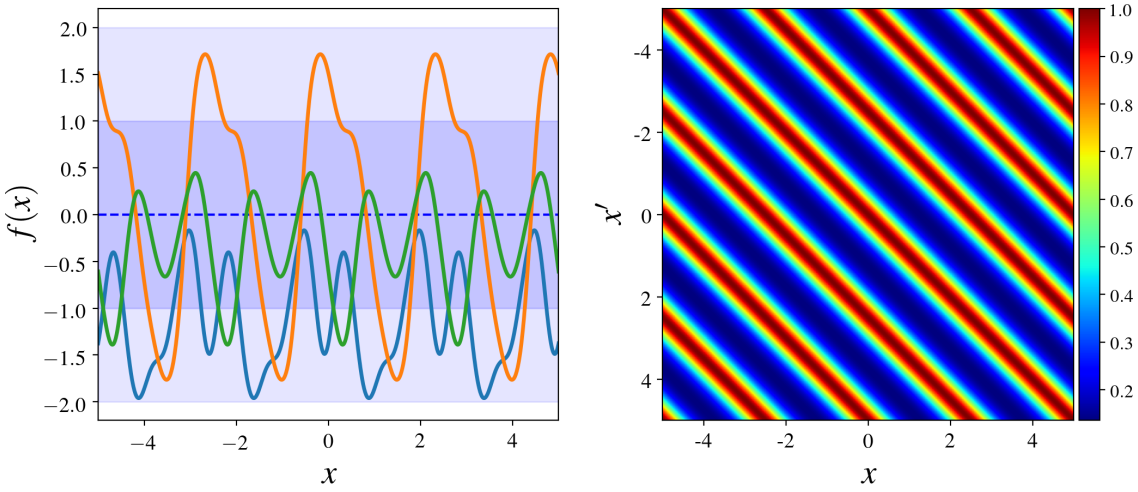

Figure 3.13: On the left, random functions drawn from a zero-mean Gaussian Process with periodic covariance function (3.46), for $\ell = 1$, $\sigma_f = 1$ and $p = 2.5$. On the right, a visual representation of the covariance matrix.

with parameter p defining the period i.e. the distance between repetitions. The meaning of parameters ℓ and σ_f is analogous to the SE covariance function (3.42). A few samples drawn for the periodic kernel function as well as the structure of the covariance matrix with $\ell = 1$, $\sigma_f = 1$ and $p = 2.5$ are shown in Figure 3.13. Note that the prior functions repeat periodically each $|x - x'| = 2.5$. This periodic structure can also be recognized in the covariance matrix, which has bands spaced by a distance of 2.5 in the input space.

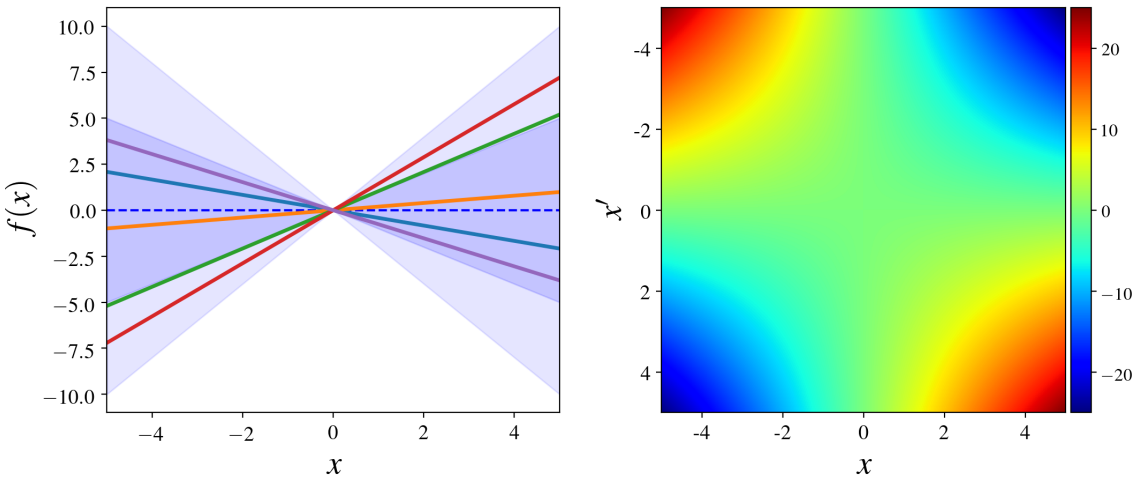

Figure 3.14: On the left, random functions drawn from a zero-mean Gaussian Process with periodic covariance function (3.47), for $\sigma_0^2 = 0$. On the right, a visual representation of the covariance matrix.

Linear Covariance Function

The linear covariance function

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_0^2 + \boldsymbol{x} \cdot \boldsymbol{x}' \tag{3.47}$$

can be obtained from linear regression by putting $\mathcal{N}(0, 1)$ priors on the coefficients and a prior of $\mathcal{N}(0, \sigma_0^2)$ on the bias. The bias σ_0^2 is the point where the function is likely to cross the vertical axis. Note that in contrast with the previous kernels it does not depend on $|\boldsymbol{x} - \boldsymbol{x}'|$. This type of covariance function is known as a dot product function since depends only on $\boldsymbol{x}$ and $\boldsymbol{x}'$ through $\boldsymbol{x} \cdot \boldsymbol{x}'$.

The behavior of the samples from the prior and the covariance matrix is shown in Figure 3.15. We can easily see that all the sample functions are linear. For regression problems the linear kernel is a rather strange choice, as we can observe, the prior variance grows rapidly with $|\boldsymbol{x}|$ for $|\boldsymbol{x}| > 1$.

3.3.2 Combining Kernels

In the previous section, we have developed many covariance functions. Now, we show how these can be combined to make new ones. The advantage is that, by combining various elemental covariance functions, we can create all sorts of more complex kernels. So if we know that the function $f(\boldsymbol{x})$ we are approximating has some kind of structure, we can take it into account. The basic operations that we discuss are the sum and the product.

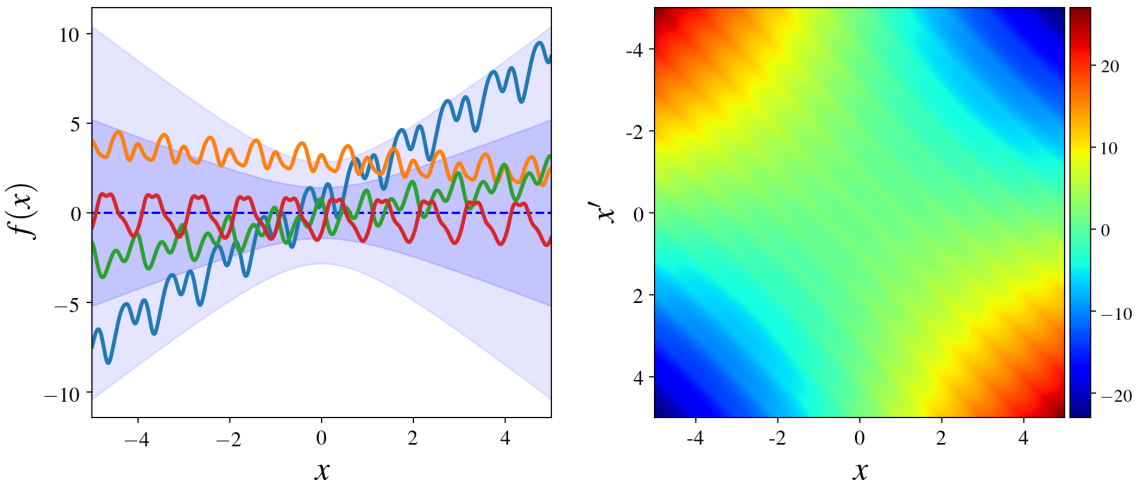

Figure 3.15: On the left, random functions drawn from a zero-mean Gaussian Process with a kernel constructed by the addition of a periodic kernel (3.46) with $\sigma_f^2 = 1$, $\ell = 1$ and $p = 1$, and a linear kernel (3.47) with $\sigma_0^2 = 0$. On the right, a visual representation of the covariance matrix.

Combining Kernels by Addition

The sum of two kernels is a kernel. Suppose that we have a function $f(\boldsymbol{x}) = f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})$. If $f_1(\boldsymbol{x})$ is a GP with mean $m_1(\boldsymbol{x})$, covariance function $k_1(\boldsymbol{x}, \boldsymbol{x}')$ and similarly for $f_2(\boldsymbol{x})$, assuming the two GPs are independent, then $f(\boldsymbol{x})$ is a GP with mean and covariance function

$$m(\boldsymbol{x}) = m_1(\boldsymbol{x}) + m_2(\boldsymbol{x}) \tag{3.48}$$

$$k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}, \boldsymbol{x}') + k_2(\boldsymbol{x}, \boldsymbol{x}') \tag{3.49}$$

Roughly speaking, adding two kernels can be interpreted as an OR operation. This means that the covariances of the two added kernels will only have a low value if both of the covariances have a low value. An application of this method is shown in Figure 3.15, where you can see the Gaussian Process resulting from the addition of a periodic kernel (3.46) and a linear kernel (3.47). The sampled functions are periodic with increasing mean as they move away from the origin.

Combining Kernels by Multiplication

The product of two kernels is a kernel. However the developement is not as simple as with the addition. Assuming that $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ are independent, then we have the following mean and variance for $f(\boldsymbol{x}) = f_1(\boldsymbol{x}) f_2(\boldsymbol{x})$,

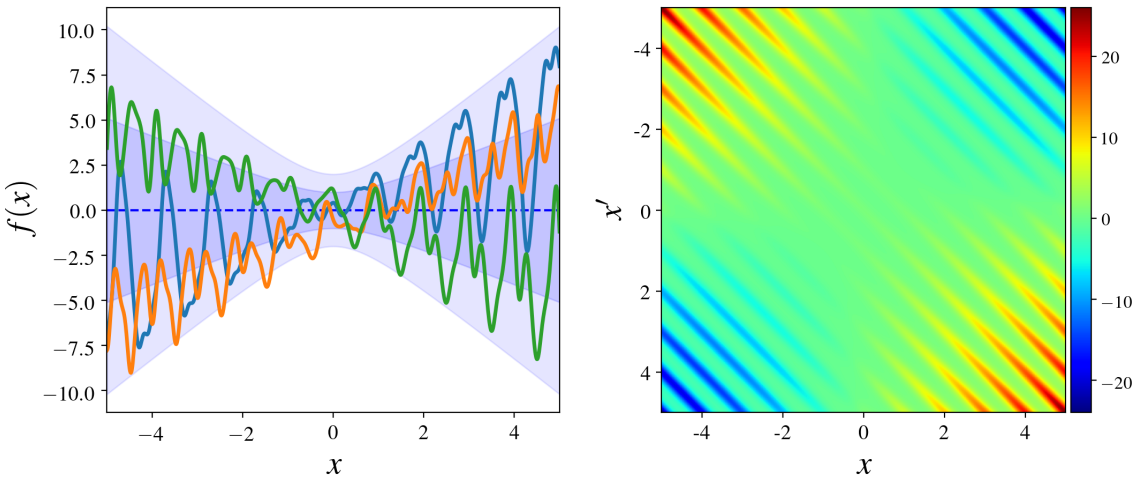

Figure 3.16: On the left, random functions drawn from a zero-mean Gaussian Process with a kernel constructed by the product of a periodic kernel (3.46) with $\sigma_f^2 = 1$, $\ell = 1$ and $p = 1$, and a linear kernel (3.47) with $\sigma_0^2 = 0$. On the right, a visual representation of the covariance matrix.

$$m(\boldsymbol{x}) = m_1(\boldsymbol{x}) \cdot m_2(\boldsymbol{x}) \tag{3.50}$$

$$\begin{aligned} k(\boldsymbol{x}, \boldsymbol{x}') = {} & k_1(\boldsymbol{x}, \boldsymbol{x}')k_2(\boldsymbol{x}, \boldsymbol{x}') + k_1(\boldsymbol{x}, \boldsymbol{x}')m_2(\boldsymbol{x})m_2(\boldsymbol{x}')+ \\ & + k_2(\boldsymbol{x}, \boldsymbol{x}')m_1(\boldsymbol{x})m_1(\boldsymbol{x}') \end{aligned} \tag{3.51}$$

If, $m_1(\boldsymbol{x}) = m_2(\boldsymbol{x}) = 0$, we can hence just use $k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}, \boldsymbol{x}')k_2(\boldsymbol{x}, \boldsymbol{x}')$. Multiplying kernels is an element-wise multiplication of their corresponding covariance matrices. This means that the covariances of the two multiplied kernels will only have a high value if both covariances have a high value. The product operation can thus be interpreted as an AND operation. An example is shown in Figure 3.16 combining, as in Figure 3.15, a linear and a periodic kernel. This results in a periodic function with linearly varying amplitude.

3.4 Hyperparameter Tuning and Model Selection

So far we have seen how to do GP regression using a given fixed covariance function. However, in many practical applications, it may not be easy to specify all aspects of the covariance function with confidence. While some properties such as stationarity may be easy to determine a priori, we typically have only rather vague information about other properties, such as the value of free (hyper-)parameters, e.g. length scales or noise level.

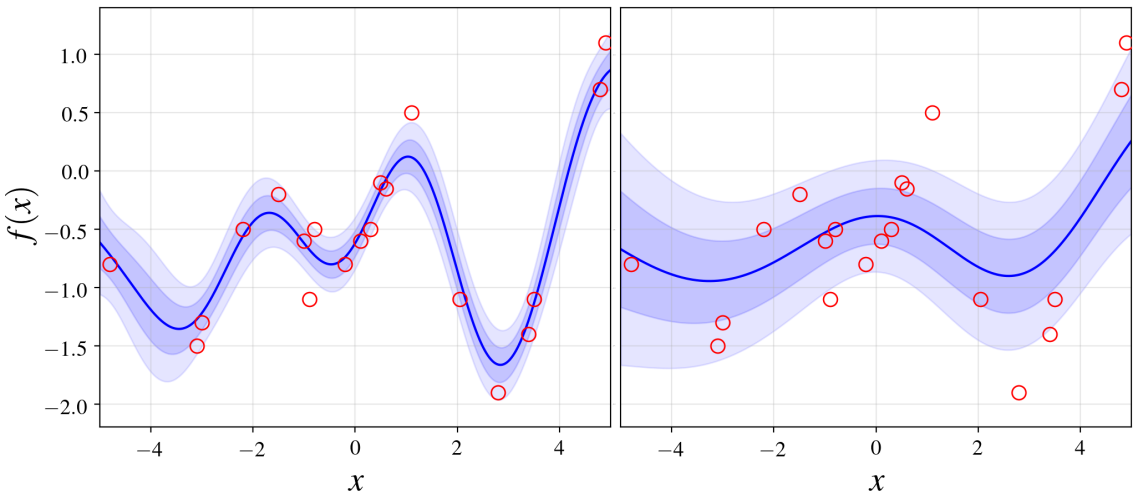

Figure 3.17: Gaussian Process regression for different hyperparameters $\left(\sigma_f^2, \ell, \sigma_n^2,\right)$ with value $(1.0, 0.5, 0.05)$ (left) and $(2.0, 0.8, 0.5)$ (right). While the first approximation seems sensible, the second explains the training data (red dots) almost through noise.

For instance, consider a GP with a prior covariance function built from the addition of (3.41) and (3.42)

$$k(x_i, x_j) = \sigma_f^2 \exp\left(-\frac{1}{2}\frac{(x_i - x_j)^2}{\ell^2}\right) + \sigma_n^2 \delta_{ij} \tag{3.52}$$

There are several hyperparameters that need to be selected, in this case $\boldsymbol{\theta} = \left(\sigma_f^2, \ell, \sigma_n^2,\right)$. In Figure 3.17, we can see that they have a significant effect on the predictions. We can choose them ourselves, based on our expert knowledge, or we can tune them automatically. In order to turn Gaussian Processes into powerful practical tools, it is essential to develop methods that address the tuning of the hyperparameters and the model selection problem.

We first discuss, in Section 3.4.1, the model selection problem from a general perspective within the probabilistic framework. Then, in Section 3.4.2, we focus on Gaussian Process regression, exploring a simple method for optimizing the covariance function hyperparameters.

3.4.1 Bayesian Model Selection

The key realization we should first make is that we do not know the set of hyperparameters $\boldsymbol{\theta}$, and hence we should treat it as a random variable. Also, we can include a higher level formed by a set of possible model structures, $\mathcal{H}_i$ under consideration. We can divide the model selection problem into a two-step process:

1. At the lower level, we can look at the probability that a certain set of hyperparameters $\boldsymbol{\theta}_i$ is the correct one for each candidate model $\mathcal{H}_i$ given the data $\mathcal{D} = (X, \boldsymbol{y})$.

2. At the higher level, we can look at the probability that a certain model $\mathcal{H}_i$ explains the data $\mathcal{D} = (X, \boldsymbol{y})$ given the distribution of $\boldsymbol{\theta}_i$.

Recalling the Bayes' theorem

$$p(A \mid B, C) = \frac{p(B \mid A, C) \cdot p(A \mid C)}{p(B \mid C)} \tag{3.53}$$

where $p(A \mid B, C)$ can be read as 'the probability of A given B and C'. The first step in the model selection process requires the computation of the posterior over the hyperparameters for a certain model $\mathcal{H}_i$. It can be written as

$$p(\boldsymbol{\theta}_i \mid \boldsymbol{y}, X) = \frac{p(\boldsymbol{y} \mid X, \boldsymbol{\theta}_i) \cdot p(\boldsymbol{\theta}_i \mid X)}{p(\boldsymbol{y} \mid X)} \tag{3.54}$$

The term $p(\boldsymbol{y} \mid X, \boldsymbol{\theta}_i)$ is known as the likelihood. It is the probability that, given certain hyperparameters $\boldsymbol{\theta}_i$, we obtained observations $\boldsymbol{y}$. According to (3.25), this probability is equal to

$$p(\boldsymbol{y} \mid X, \boldsymbol{\theta}_i) = \mathcal{N}\left(\boldsymbol{m} \mid \boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right) \tag{3.55}$$

Note that the covariance $\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}$ and possibly the mean $\boldsymbol{m}$ depend on the hyperparameters $\boldsymbol{\theta}_i$. The term $p(\boldsymbol{\theta}_i \mid X)$ is the prior hyperparameter distribution, or short, the hyper-prior. This probability actually does not depend on X since only knowing X does not tell us anything about $\boldsymbol{\theta}_i$. Thus, $p(\boldsymbol{\theta}_i \mid X) \equiv p(\boldsymbol{\theta}_i)$. We can use the hyper-prior to indicate which hyperparameters we roughly expect to get. In practice, we often don't really know much in advance so $p(\boldsymbol{\theta}_i)$ is assumed constant. Finally, the denominator $p(\boldsymbol{y} \mid X)$ is called the marginal likelihood, but since it does not depend on $\boldsymbol{\theta}_i$, it is a constant too. Putting all together we have that the probability of $\boldsymbol{\theta}_i$ is proportional to the likelihood

$$p(\boldsymbol{\theta}_i \mid \boldsymbol{y}, X) \propto p(\boldsymbol{y} \mid X, \boldsymbol{\theta}_i) \tag{3.56}$$

Then, at the second level of the model selection process, we evaluate the posterior of the model

$$p(\mathcal{H}_i \mid \boldsymbol{y}, X) = \frac{p(\boldsymbol{y} \mid X, \mathcal{H}_i) \cdot p(\mathcal{H}_i \mid X)}{p(\boldsymbol{y} \mid X)} \tag{3.57}$$

Following the same reasoning as with the hyperparameter posterior, and marginalizing over the hyperparameters, we have

$$p(\mathcal{H}_i \mid \boldsymbol{y}, X) \propto p(\boldsymbol{y} \mid X, \mathcal{H}_i) = \int p(\boldsymbol{y} \mid X, \boldsymbol{\theta}_i, \mathcal{H}_i) \cdot p(\boldsymbol{\theta}_i \mid \mathcal{H}_i) d\boldsymbol{\theta}_i \tag{3.58}$$

In practice, the evaluation of (3.58) may be difficult, and as an approximation, the following is usually assumed

$$p(\mathcal{H}_i \mid \boldsymbol{y}, X) \propto p(\boldsymbol{y} \mid X, \boldsymbol{\theta}_i) \tag{3.59}$$

Therefore, we can conclude that the model selection problem can be reduced to the evaluation of the likelihood with respect to the possible hyperparameter sets $\boldsymbol{\theta}_i$.

3.4.2 Model Selection for GP Regression

Now, we apply the general Bayesian inference principles from subsection 3.4.1 to the specific Gaussian Process model. Instead of taking into account all possible hyperparameters $\boldsymbol{\theta}$, the idea is to find the most likely hyperparameters and only use those. The key is to maximize the likelihood $p(\boldsymbol{y} \mid X, \boldsymbol{\theta})$

$$p(\boldsymbol{y} \mid X, \boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi |\boldsymbol{K}_y|}} \exp\left(-\frac{1}{2} (\boldsymbol{y} - \boldsymbol{m})^T \boldsymbol{K}_y^{-1} (\boldsymbol{y} - \boldsymbol{m})\right) \tag{3.60}$$

The exponent makes maximizing this somewhat difficult. To solve this issue, we take the logarithm of the above function. Because the logarithm is a strictly ascending function this does not affect the position of the maximum. When we take the logarithm and simplify the result, we get the log-likelihood

$$\log p(\boldsymbol{y} \mid X, \boldsymbol{\theta}) = -\frac{1}{2} (\boldsymbol{y} - \boldsymbol{m})^T \boldsymbol{K}_y^{-1} (\boldsymbol{y} - \boldsymbol{m}) - \frac{1}{2} \log |\boldsymbol{K}_y| - \frac{n}{2} \log 2\pi \tag{3.61}$$

where $\boldsymbol{K}_y = \boldsymbol{K} + \sigma_n^2 \boldsymbol{I}$ is the covariance matrix for the noisy targets $\boldsymbol{y}$. The three terms of the marginal likelihood in (3.61) have readily interpretable roles. The only term involving the observed targets is the data-fit $(\boldsymbol{y} - \boldsymbol{m})^T \boldsymbol{K}_y^{-1} (\boldsymbol{y} - \boldsymbol{m}) /2$, which describes how well our measurements $\boldsymbol{y}$ fit with our hyperparameters. The second term $|\boldsymbol{K}_y|/2$ is the complexity penalty depending only on the covariance function and the inputs. The great thing about this complexity penalty is that, unlike other methods like neural networks, Gaussian process regression has a far smaller risk of overfitting. It has an automatic regularization built into it through its foundation in Bayesian probability theory. Finally, $n \log (2\pi)/2$ is just a normalization constant.

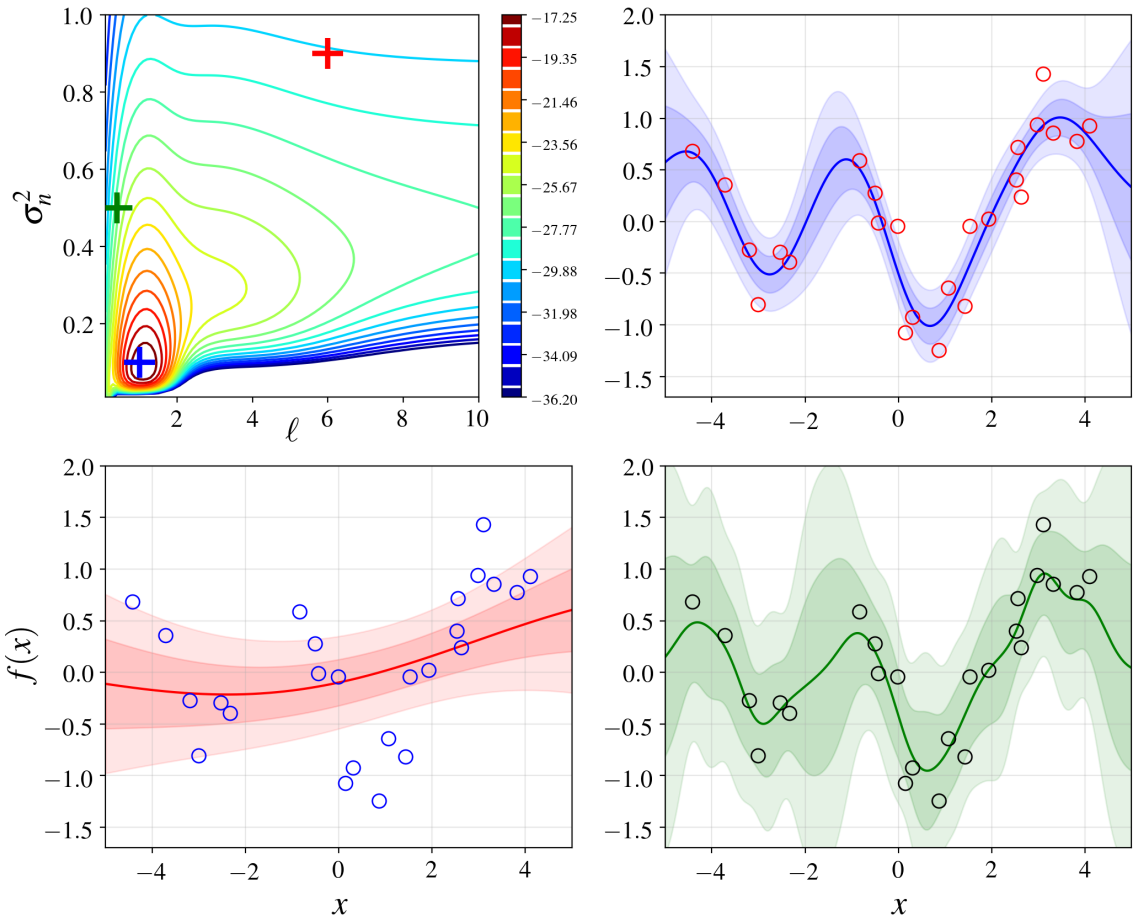

Figure 3.18: On the top left contour plot showing the log marginal likelihood for data generated using a GP with $(\sigma_f^2, \ell, \sigma_n^2) = (1.0, 1.0, 0.1)$ (3.52) as a function of the characteristic length-scale l and the noise level σ_n^2. On the remaining plots, three GPs with $\sigma_f^2 = 1$, and ℓ and σ_n^2 marked as a cross with the corresponding color (blue, red, and green) in the contour plot. We can see that the most sensible option among the candidates is also the one with the highest log-likelihood.

Figure 3.18 shows an example of the log marginal likelihood as a function of the characteristic length-scale and the noise level hyperparameters for covariance function (3.52). The signal variance σ_f^2 is set to 1. The marginal likelihood has a clear maximum around the hyperparameter values which were used in the Gaussian Process from which the data was generated. Note that for length scales and a noise level higher than $\sigma_n^2 = 0.8$, the marginal likelihood becomes almost independent of the length scale; this is caused by the model explaining everything as noise, and no longer needing the signal covariance. Similarly, for small noise and small length scales, the marginal likelihood becomes almost independent of the noise level; this is caused by the ability of the model to exactly interpolate the data at this short length scale.

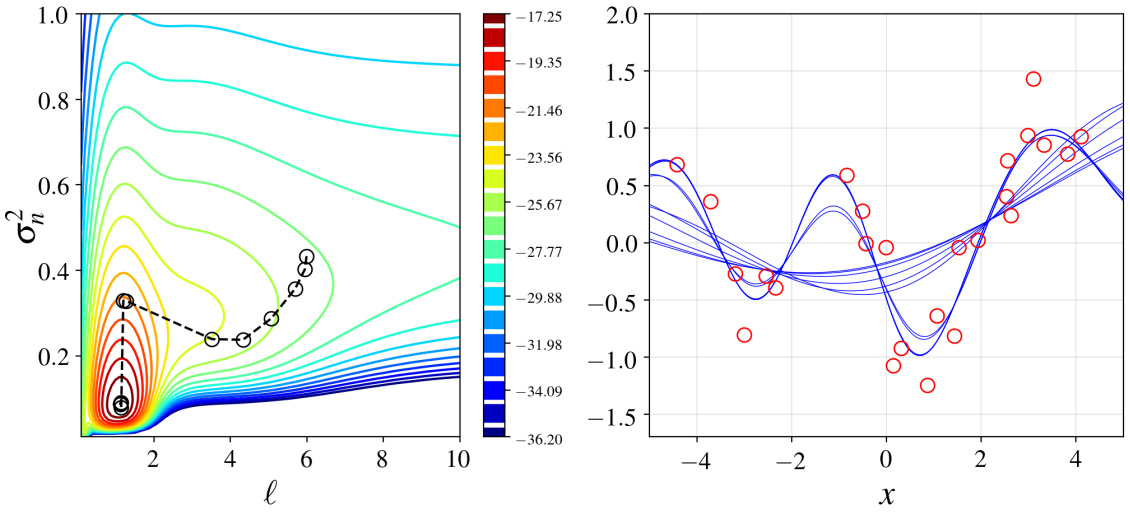

Figure 3.19: On the right, the mean of the prediction while tuning the hyperparameters. We use the same data as Figure 3.18 but start with hyperparameters $(\sigma_f^2, \ell, \sigma_n^2) = (1.0, 6.0, 8)$. We then optimize them using a gradient ascent algorithm, converging to $(\sigma^2, \ell, \sigma_n^2) = (1.0, 1.124, 0.087)$. The steps of the gradient ascent can be seen on the contour plot of the likelihood on the left.

The next question is: how do we find the maximum of the log-likelihood? There are many optimization methods to do so. A common choice is the gradient ascent method for which the partial derivatives of the marginal likelihood with respect to the hyperparameters are required. Differentiating (3.61) we obtain

$$\frac{\partial}{\partial \theta_j} \log p(\boldsymbol{y} \mid X, \boldsymbol{\theta}) = \frac{1}{2} \operatorname{tr}\left(\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^T - \boldsymbol{K}_y \right) \frac{\partial \boldsymbol{K}_y}{\partial \theta_j} \right) \tag{3.62}$$

where $\boldsymbol{\alpha} = \boldsymbol{K}_y^{-1}(\boldsymbol{y} - \boldsymbol{m})$. Note that the term $\partial \boldsymbol{K}_y / \partial \theta_j$ depends on the covariance function. An illustration of the evolution of the GP prediction, adapting the hyperparameters with the gradient ascent algorithm is shown in Figure 3.19.

The challenge of the Gaussian Process selection problem is that there is no guarantee that the marginal likelihood does not suffer from multiple local optima. The gradient ascent algorithm is not suitable for these cases. Thus, more complex, but standard, optimization algorithms can be used to determine the hyperparameters. An overview of some alternative optimization methods based on evolutionary algorithms is provided in [7].

Part II

Robot Learning from Demonstration

Chapter 4

Related Work

As robots move from structured and simple problems to more complex, unstructured environments, manually programming their behavior is requiring a more specific set of skills and knowledge. Often, it is much easier for a human to demonstrate the desired behavior rather than attempt to engineer it. This is the main behind the paradigm of learning from demonstration (LfD).

Learning from demonstration may be a key technology to shift robots from industrial applications to work closely with humans in elder care or the service industry. Classical industrial robotic manipulators are powerful, but also dangerous. For this reason, they have been used mainly in constrained, repetitive tasks. In recent years this is changing due to the development of light, compliant and safe robotic manipulators. They are ideal for applications that benefit from human-robot collaboration, such as reducing the physical workload of caregivers. These applications need efficient and intuitive ways to teach robots the required motions.

Over the last decade, learning from demonstration has been an intensive field of study, for which research interest has done nothing but steadily increase. As it can be seen in Figure 4.1, the number of publications related to the topic during 2020 is almost four times larger than in 2010. Also note that, although we use the term learning from demonstration to encompass the field as a whole, other popular terms are used in the literature such as *imitation learning*, *programming by demonstration*, and *behavioral cloning*, among others.

Different learning approaches, namely supervised, reinforcement, and unsupervised, have been used to address a plethora of problems in robot learning. The choice between the different methods is not trivial and depends on the problem of interest. From a general perspective, to allow robots to learn skills from human demonstrations, we need to develop a system that records demonstrations by experts, learns the ideal behavior from the available demonstrations, and reproduces it.

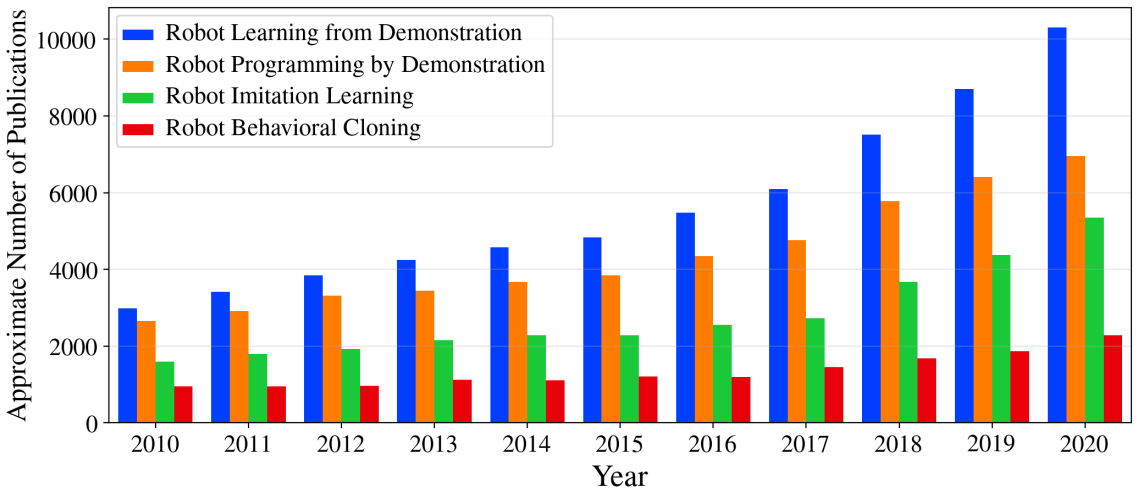

Figure 4.1: Evolution of the estimated number of publications during the last decade in the field of learning from demonstration (LfD). The estimate is obtained as the number of search results on Google Scholar that contain the specified key terms related to LfD.

For addressing the learning from demonstration problem we need to answer the following questions [8]:

1. **How should we record the data of the expert demonstrations?** There is a wide variety of methods for recording human demonstrations. Some examples are motion capture or teleoperation systems. This choice depends on the differences in the embodiment between the human teacher and the robot learner. In Chapter 5, we discuss this question in detail.

2. **What should we imitate?** The recorded data might include redundant information about the ideal behavior, or unnecessary motions, which should not be imitated. We must carefully select the appropriate features.

3. **How should we represent the skill?** The ideal behavior can be represented using methods such as trajectory-based representation or state-action space representation. The choice depends largely on the level of abstraction that is appropriate for the problem of interest.

4. **How should we learn the ideal behavior from the demonstrations?** Many algorithms for learning the demonstrated skill have been developed over the last years. The choice of the most suitable algorithm is closely related to the previous question.

With regard to these questions, several survey papers on robot learning from demonstration provide a distinct overview of the field by answering them from different perspectives [8, 9, 10, 11].

In order to keep the discussion within the scope of a Chapter, we concentrate on the last two questions and focus on reviewing the state-of-the-art learning trajectories from demonstrations. This family of algorithms is one of the most dominant in LfD research. Trajectory learning methods rely on low-level controllers to execute the trajectories required to perform the taught skill. They are so popular because, if we assume that the system is fully actuated, which is the case for most robot manipulators, we do not need any knowledge of the robot dynamics.

We start in Section 4.1, by presenting the fundamental aspects of Dynamic Movement Primitives (DMPs). Then, in Section 4.2, we discuss the main concepts behind Gaussian Mixture Models (GMMs). Afterward, in Section 4.3, we study some of the most relevant features of Probabilistic Movement Primitives (ProMPs). And finally, in Section 4.4, we introduce the Kernelized Movement Primitives (KMPs) formulation.

4.1 Dynamic Movement Primitives

Human beings have the innate ability to perform complex manipulation tasks in a versatile manner. Researchers have tried to understand and define this capability, concluding that is based on combining and adapting basic units of motion into complex movements. This is the origin of the motion primitives theory.

Dynamic Movement Primitives (DMPs) represent one of the first attempts to design a method that allows robots to execute movements in a versatile and adaptive manner. They can be seen as a formal mathematical formulation of the motion primitives as stable nonlinear dynamical systems [12]. These systems must guarantee convergence to a given target, be sufficiently flexible to create complex behaviors, and possible to learn from demonstrations using efficient algorithms.

DMPs have been successfully exploited in a wide variety of applications. Due to their simple formulation, they have become the first approach that beginners in learning from demonstration use on their robots. For the interested reader, we refer to the following tutorials and surveys, [13, 14, 15, 16]. Especially [16], since it is the most recent one and scans a wider spectrum within the Dynamic Movement Primitives related literature.

Here, we provide an overview of Dynamic Movement Primitives. First, in Section 4.1.1, we introduce the DMPs formulation for discrete and periodic motions. Then, in Section 4.1.2, we discuss some extensions of the DMP formalism for improving the generalization capabilities of the learned skill or combine multiple primitives. Finally, in Section 4.1.3, we present DMPs in different robotics applications.

4.1.1 Basic Formulation

In this section, we introduce the standard formulations of DMPs. Specifically, to learn discrete point-to-point motions and rhythmic pattern motions.

Discrete DMPs

The basic idea behind a classical discrete DMP, is to use a dynamical system with convenient stability properties and modulate it with nonlinear terms such that it achieves the desired attractor behavior. One of the simplest possible models for a single-degree-of-freedom trajectory y, is a damped spring model, defined by the following set of nonlinear equations

$$\tau \dot{z} = \alpha_z \left(\beta_z \left(g - y \right) - z \right) + f(x) \tag{4.1}$$

$$\tau \dot{y} = z \tag{4.2}$$

$$\tau \dot{x} = -\alpha_x x \tag{4.3}$$

Here, z is known as the auxiliary variable, the parameter $\tau > 0$ is a time constant, and $\alpha_z > 0$ and $\beta_z > 0$ define the behavior of the system described by equations (4.1) and (4.2), commonly referred to as the transformation system. On the other hand, equation (4.3) defines the so-called canonical system, being x the phase variable and $\alpha_x > 0$ a positive constant.

The key to achieve the desired behavior with DMPs, is the function $f(x)$, which is commonly defined as a linear combination of N nonlinear radial basis functions

$$f(x) = \frac{\sum_{=1}^{N} w_i \phi_i(x)}{\sum_{i=1}^{N} \phi_i(x)} \cdot x \left(g - y_0 \right) \tag{4.4}$$

$$\phi_i(x) = \exp\left(-\frac{1}{2\ell_i^2} \left(x - \mu_i \right)^2 \right) \tag{4.5}$$

Here, μ_i are the centers of the basis functions distributed along with the phase of the movement, ℓ_i their widths, and y_0 the initial position. Note that the complete system is designed to have a unique equilibrium point at $(x, y, z) = (0, g, 0)$. Thus, a DMP serves as a basis for generating a discrete movement for y, evolving towards the goal g, from any initial position y_0.

We assume that skill is demonstrated by providing a desired trajectory in terms of position, velocity, and acceleration triples

$$\mathcal{D} = \{ (y_d(t_j), \dot{y}_d(t_j), \ddot{y}_d(t_j)) \}_{j=1}^{M} \tag{4.6}$$

Learning the taught behavior is performed in two phases: determining the high-level parameters (g, y_0 and τ) and then learning the weights w_i. The parameter g is simply the position at the end of the demonstration $g = y_d(t_M)$, and analogously, $y_0 = y_d(t_1)$. The time constant τ must be adjusted to the duration of the demonstration.

Then, the learning of the weights can be formulated as a regression problem. Rearranging equation (4.1), and including the information provided by the demonstrated trajectory (4.6), we have

$$f_d(t_j) = \tau^2 \ddot{y}_d(t_j) - \alpha_z \left(\beta_z \left(g - y_d(t_j)\right) - \tau \dot{y}_d(t_j)\right) \tag{4.7}$$

Therefore, we have to adjust the weights w_i such that $f(x)$ is as close as possible to $f_d(t_j)$. There are many possible methods for solving this problem. The most common approach is to use locally weighted regression (LWR) [17] due to its efficient one-shot learning procedure and the fact that each weight is learned independently. For each w_i, LWR minimizes the local weighted quadratic error

$$E(w_i) = \sum_{j=1}^{M} \phi_i \left(x(t_j)\right) \left(f_d(t_j) - w_j \zeta(t_j)\right)^2 \tag{4.8}$$

with $\zeta(t_j) = x(t_j)\,(g - y_0)$. This optimization problem has the following closed-form solution

$$w_i^* = \frac{\boldsymbol{\zeta}^T \boldsymbol{\Phi}_i \boldsymbol{f}_d}{\boldsymbol{\zeta}^T \boldsymbol{\Phi}_i \boldsymbol{\zeta}} \tag{4.9}$$

where

$$\boldsymbol{\zeta} = \begin{bmatrix} \zeta(t_1) \\ \zeta(t_2) \\ \vdots \\ \zeta(t_M) \end{bmatrix}, \quad \boldsymbol{\Phi}_i = \begin{bmatrix} \phi_i\left(x(t_1)\right) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \phi_i\left(x(t_M)\right) \end{bmatrix}, \quad \boldsymbol{f}_d = \begin{bmatrix} f_d(t_1) \\ f_d(t_2) \\ \vdots \\ f_d(t_M) \end{bmatrix} \tag{4.10}$$

An example of a DMP learned after providing a synthetic demonstration is shown in Figure 4.2. We can observe that the demonstrated trajectory and the resulting DMP are almost perfectly superposed. Also, we can see the sequential activation of the basis functions comprising the forcing term $f(x)$.

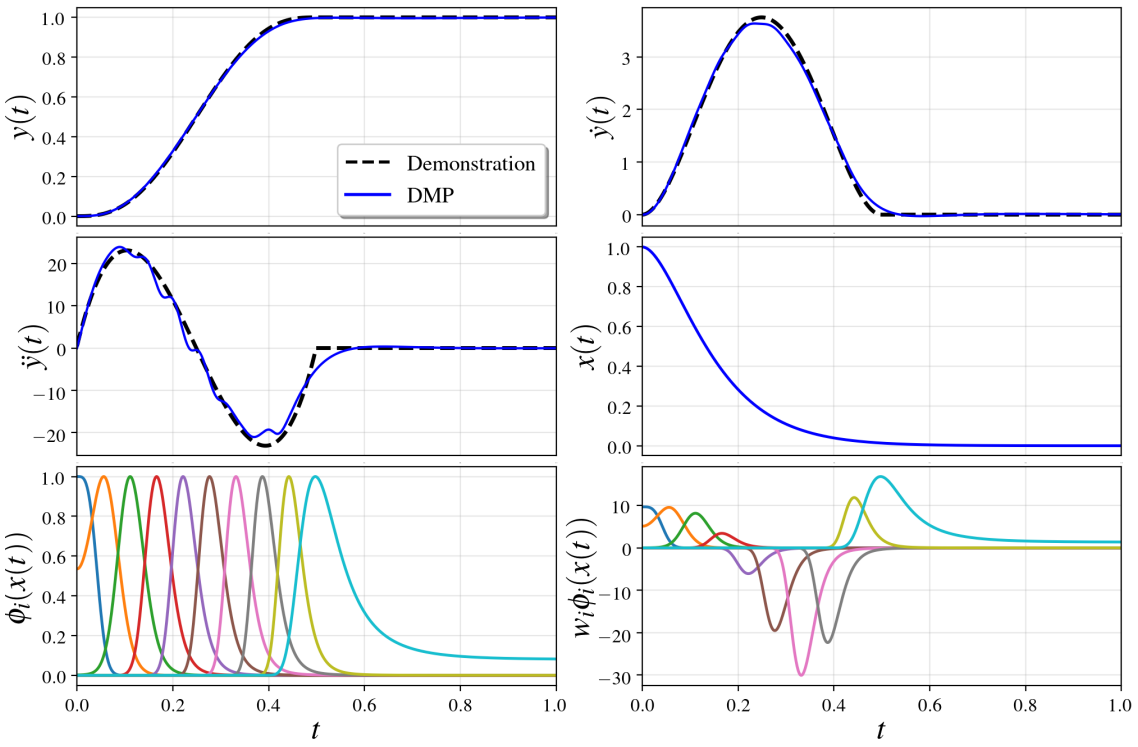

Figure 4.2: A discrete DMP is used to generate 1-dimensional motion between $y_0 = 0$ and $g = 1$. The learned trajectory (blue solid line) along with the demonstration data (black dashed line) is shown in the three upper left graphs. On the center-right, the exponentially decaying phase. The plots on the bottom row illustrate the results of LWR.

Periodic DMPs

Periodic DMPs [18] are used when the motion follows a rhythmic pattern. The second order system described by (4.1), (4.2) and (4.3) is redefined as follows

$$\dot{z} = \Omega\left(-\alpha_z\left(\beta_z y + z\right) + f(\psi)\right) \tag{4.11}$$
$$\dot{y} = \Omega z \tag{4.12}$$
$$\tau\dot{\psi} = 1 \tag{4.13}$$

The main difference between discrete and periodic DMPs is that the time constant τ, related to the trajectory duration, is replaced by the frequency Ω. Analogously to (4.4), the forcing term is defined as

$$f(\psi) = \frac{\sum_{i=1}^{N}\phi_i(\psi)w_i}{\sum_{i=1}^{N}\phi_i(\psi)} \cdot r \tag{4.14}$$
$$\phi_i(\psi) = \exp\left(h_i\left(\cos\left(\psi - \mu_i\right) - 1\right)\right) \tag{4.15}$$

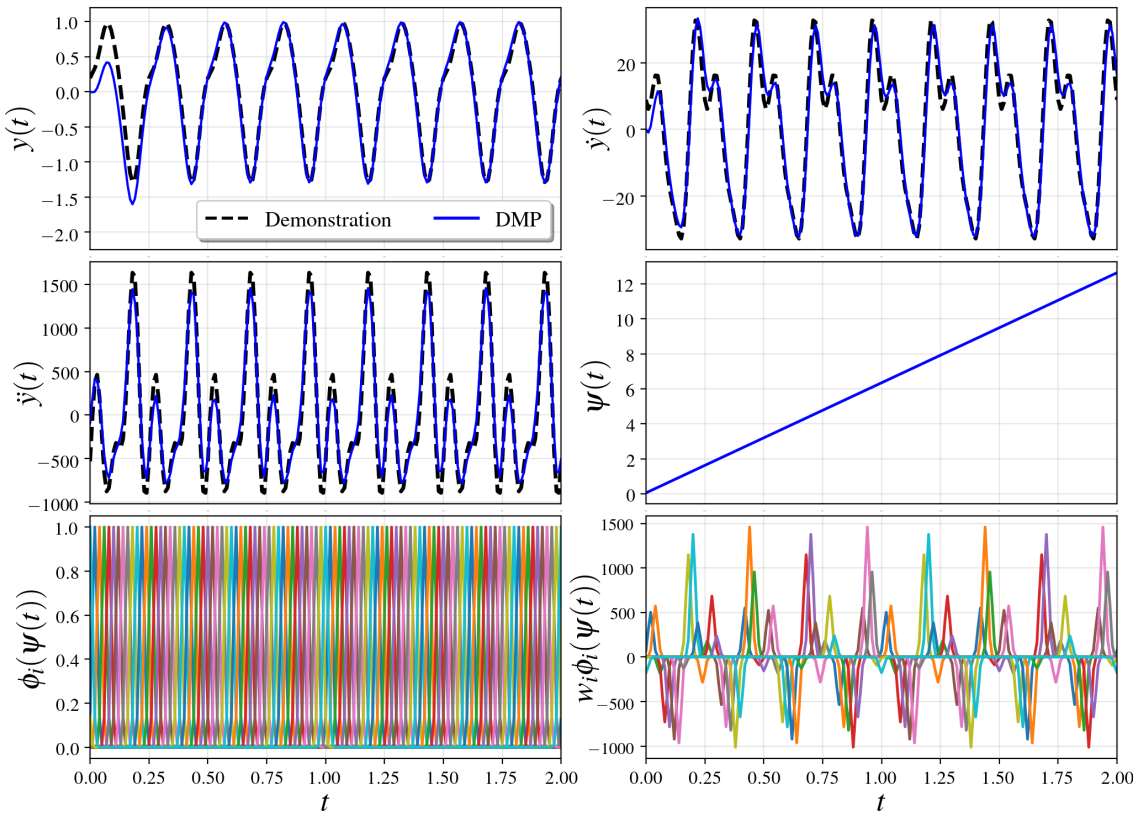

Figure 4.3: A periodic DMP is used to reproduce a 1-dimensional rhythmic motion. The learned trajectory (blue solid line) along with the demonstration data (black dashed line) is shown in the three upper left graphs. The plots on the bottom row illustrate the weights learning procedure.

where r is used to modulate the amplitude of the periodic signal, and h_i is a scale parameter analogous to ℓ in (4.5). Similarly to discrete DMPs, the desired shape of $f(\psi)$ is determined as follows

$$f_d(t_j) = \frac{\ddot{y}_d(t_j)}{\Omega^2} + \alpha_z \left(\beta_z y_d(t_j) + \frac{\dot{y}_d(t_j)}{\Omega} \right) \tag{4.16}$$

Then, the learning of the weights can also be performed with LWR taking $\zeta(t_j) = r$, and using equations (4.9) and (4.10).

In Figure 4.3, we show an example of a periodic DMP, when trained with a superposition of several sine signals of different frequencies. Note how quickly the DMP converges to the demonstrated trajectory from zero initial conditions. The evolution of the velocity and the acceleration is also almost perfectly coincident with the taught ones.

4.1.2 Extensions

In this section, we present approaches that allow adapting skills learned with DMPs to novel situations, and how to smoothly combine several simple DMPs to obtain a more complex behavior.

Generalization of a DMP

A desirable property of motion primitives is the ability to generalize the learned skill to unforeseen scenarios. One possibility to achieve such capability is to modulate the DMP through via-points (points where the trajectory has to pass). Weitschat et al. [19] considered each via-point as an intermediate goal to reach g_v for $v = 1, \ldots, V$. Then, they redefined the constant goal in g (4.1) as a variable in the following way

$$g(x) = \sum_{v=1}^{V} \phi_v(x) g_v \tag{4.17}$$

where $\phi_v(x)$ are radial basis function centered at the time corresponding to the v-th via-point. By changing the goal position, the DMP maintains the general shape of the learned trajectory but is adapted to pass through the via-points.

Sometimes, modulating the DMP using via-point constraints may not be enough to successfully perform a task in a different context. For this reason, other approaches adapt the DMP by adjusting the weights w_i of the forcing term (4.4). Weitschat et al. [20], assumed that several demonstrations are given covering the possible contexts. Each one is then encoded with a different DMP. In order to generalize to new scenarios, they proposed to interpolate the weights of DMPs encoding demonstrations in a nearby context. For instance, let us consider the case of reaching a new goal g'. The weights of the resulting DMP are computed as

$$w'_i = \frac{\sum_{\forall k: d_k < d_{\max}} w_{ik} d_k^{-1}}{\sum_{\forall k: d_k < d_{\max}} d_k^{-1}} \tag{4.18}$$

where d_k is the distance between the goal of the k-th DMP g_k and g', representing k the indices of the nearby DMPs.

In Figure 4.4, we show some examples of DMPs generalized by considering both, via-point constraints and adjusting the weights by interpolating DMPs learned for a similar context. We can see that by considering via-points the shape of the resulting trajectory is preserved except in those regions close to the new constraints. On the other hand, note that by combining the weights of several DMPs we are able to generate intermediate behaviors.

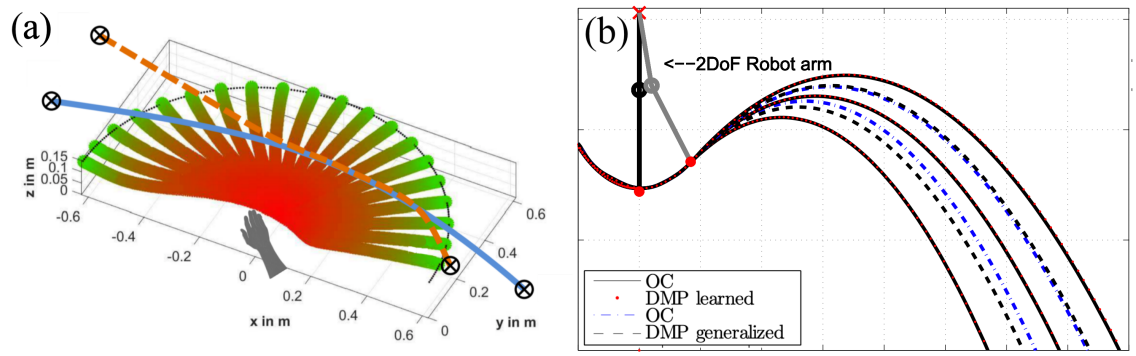

Figure 4.4: Examples of generalization of DMPs. (a) The blue solid line depicts the original DMP, and the orange dashed line the modulated trajectory using an initial and a final via-point (after [19]). (b) The black dashed line shows the DMPs interpolated from those depicted as red dotted lines. The trajectories labeled as OC in the legend are not a matter of interest in this work (after [20]).

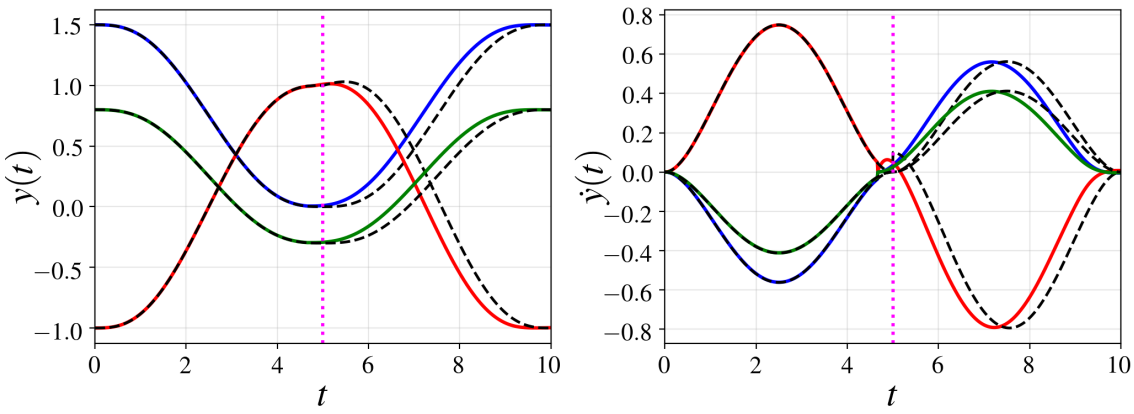

Figure 4.5: Example of a combination of DMPs. The demonstrations are represented as black dashed lines and each learned DMP as a solid line of a different color. The magenta dotted line indicates the time instant when the switch occurs (after [16]).

Combination of DMPs

An important property of a motion primitive representation is the possibility to combine several basic movements to perform complex tasks [21]. A simple approach for this purpose is proposed by Pastor et al. [22]. In general, a DMP reaches the desired target with zero velocity and acceleration. This also implies that, once the robot is close to the target, the velocity is continuously decreasing. Thus, DMPs can be combined sequentially by terminating the current DMP when the velocity is below a certain threshold and then starting the next one. To prevent discontinuities, the state of each DMP is initialized with that of the previous one. In Figure 4.5, we illustrate the combination of several pairs of DMPs.

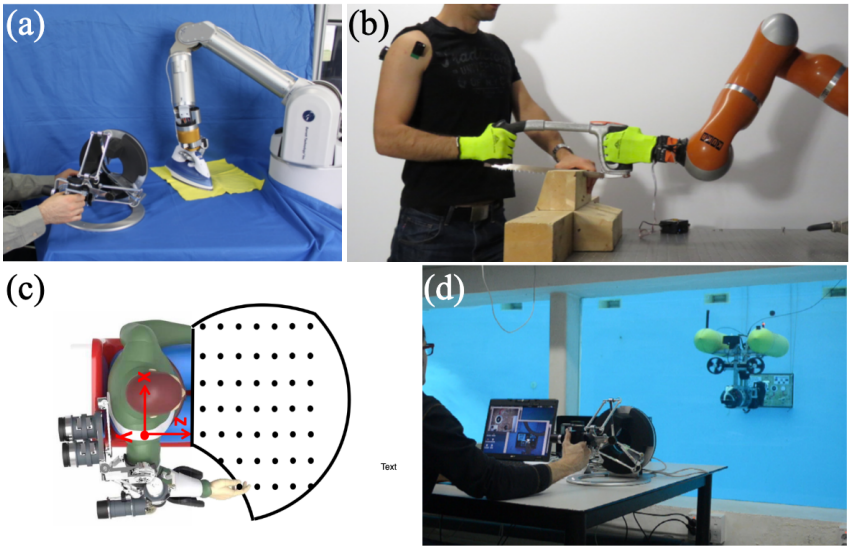

Figure 4.6: Applications of DMPs. (a) Ironing task [23]. (b) Sawing task [24]. (c) Exoskeleton for rehabilitation [25]. (d) Underwater valve turning task [26].

4.1.3 Applications

Dynamic Motion Primitives have been adopted in a wide range of applications in robotics. Most of the tasks that the robots must perform involve physical interaction with the environment, which requires the control of forces and positions. In [23], the human first guides the robot arm for teaching the position and orientation trajectories required to perform the ironing task. In the second stage, the corresponding forces are recorded with a haptic device in a teleoperation setup (Figure 4.6a). The position and force profiles are then encoded as a mixture of DMPs.

In some cases a passive interaction is not enough, the robot might have to interact with an active agent. In human-robot collaboration tasks, [27] the robot must be able to control complex movements in coordination with a human co-worker. DMPs offer an elegant solution to encode such motions. In [24], the robot is taught how to perform the collaborative sawing task (Figure 4.6b). During the execution, the robot learns the motion and impedance trajectories online using DMPs.

Another type of co-manipulation occurs when a human is wearing an exoskeleton. Usually, the exoskeleton simply amplifies human motion [28]. However, in some cases, the exoskeleton executes pre-defined trajectories during physical therapy on patients. In [25], the authors provide demonstrations of a series of movements and encode them with DMPs. Then, they use the learned motion in an arm exoskeleton for supporting the patient's movements (Figure 4.6c).

Finally, DMPs can also be applied in different autonomous mobile robots such as Autonomous Underwater Vehicles (AUVs). In [26], the authors used DMPs to teach an AUV the underwater valve turning task from demonstrations (Figure 4.6d).

4.2 Gaussian Mixture Models

Robot motions can be learned from demonstrated trajectories using deterministic methods, such as Dynamic Motion Primitives, or probabilistic methods, such as Gaussian Mixture Models (GMMs) [29]. In the latter case, the demonstrated trajectories are assumed to represent samples of a stochastic process, and thus reproductions are obtained by sampling the learned distribution over trajectories.

The theoretical analysis of GMMs and the development of related learning algorithms have been largely studied in machine learning [30]. In the context of learning from demonstration, GMMs are used to encode the joint distribution of the temporal and spatial components of continuous trajectories. Then, through Gaussian Mixture Regression (GMR) [31], we can obtain the spatial variables as a function of the temporal ones from the joint distribution.

Gaussian Mixture Models have a proven record of success in LfD [32, 33]. Most of the literature related to the field can be attributed to Sylvain Calinon. For this reason, we refer the reader interested in a more in-depth exploration of GMMs within robotics to his personal webpage [34], where open-source code and many relevant publications are available. Also, [35] summarizes in a comprehensive manner the most important ideas behind GMMs, and is the manuscript on which some of the contents of this section are based.

Here, we outline the main concepts of Gaussian Mixture Models in the context of learning from demonstration. We start, in Section 4.2.1, by reviewing the standard formulation of a combined GMM-GMR model. Next, in Section 4.2.2, we present Task-parametrized Gaussian Mixture Models (TP-GMMs). Finally, in Section 4.2.3, we discuss some relevant robotic applications.

4.2.1 Basic Formulation

The combination of GMMs with GMR allows learning smooth trajectories from the taught motions through Gaussian conditioning. Let us assume that we have a dataset of n observations $\mathcal{D} = \{\boldsymbol{\xi}_j\}_{j=1}^{n}$, with $\boldsymbol{\xi}_j \in \mathbb{R}^m$. We, therefore, consider that the dataset is distributed according to a linear combination of K Gaussian densities, which can be expressed as follows

$$p(\boldsymbol{\xi}_j) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}\left(\boldsymbol{\xi}_j \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) \tag{4.19}$$

where $\mathcal{N}(\boldsymbol{\xi}_j \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ refers to the conditional probability of $\boldsymbol{\xi}_j$

$$\mathcal{N}(\boldsymbol{\xi}_j \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{\sqrt{(2\pi)^m \boldsymbol{\Sigma}_k}} \exp\left(-\frac{1}{2}(\boldsymbol{\xi_j} - \boldsymbol{\mu_k})^T \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{\xi_j} - \boldsymbol{\mu_k})\right) \tag{4.20}$$

The GMM parameters are then $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$, being π_k the mixing coefficients, μ_k the mean, and $\boldsymbol{\Sigma}_k$ the covariance of the k-th Gaussian component. Note that if we integrate both sides of (4.19), it yields

$$\sum_{k=1}^{K} \pi_k = 1, \qquad 0 \leq \pi_k \leq 1 \tag{4.21}$$

One way to set the values of the model parameters is to maximize the likelihood of having observed the demonstration dataset

$$\{\pi_k^*, \boldsymbol{\mu}_k^*, \boldsymbol{\Sigma}_k^*\}_{k=1}^K = \underset{\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K}{\arg\max} \sum_{j=1}^{n} \log\left(\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(\boldsymbol{\xi}_j \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \tag{4.22}$$

Although several methods exist for solving this optimization problem [36, 37], the usual approach is based on an expectation-maximization (EM) procedure [38]. It is an iterative algorithm that, after initializing the parameters, alternates between the two following steps:

1. **E-step**: Evaluate the responsibility $\gamma_{j,k}$ of each component

$$\gamma_{j,k} = \frac{\pi_k \cdot \mathcal{N}(\boldsymbol{\xi}_j \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(\boldsymbol{\xi}_j \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \tag{4.23}$$

2. **M-step**: Update the estimate of the parameters

$$\boldsymbol{\mu}_k = \frac{1}{\sum_{j=1}^{n} \gamma_{j,k}} \sum_{j=1}^{n} \gamma_{j,k} \cdot \boldsymbol{\xi}_j \tag{4.24}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_{j=1}^{n} \gamma_{j,k}} \sum_{j=1}^{n} \gamma_{j,k}(\boldsymbol{\xi}_j - \boldsymbol{\mu}_k)(\boldsymbol{\xi}_j - \boldsymbol{\mu}_k)^T \tag{4.25}$$

$$\pi_k = \frac{1}{n} \sum_{j=1}^{n} \gamma_{j,k} \tag{4.26}$$

These two steps are repeated until convergence, i.e., until the difference of the log-likelihood between two consecutive iterations falls below a given threshold.

Now it is convenient decompose the demonstration data $\boldsymbol{\xi}_j$, the mean $\boldsymbol{\mu}_k$, and the covariance $\boldsymbol{\Sigma}_k$ in terms of the input (i) and output (o) dimensions

$$\boldsymbol{\xi}_j = \begin{bmatrix} \boldsymbol{\xi}_j^{(i)} \\ \boldsymbol{\xi}_j^{(o)} \end{bmatrix}, \quad \boldsymbol{\mu}_k = \begin{bmatrix} \boldsymbol{\mu}_k^{(i)} \\ \boldsymbol{\mu}_k^{(o)} \end{bmatrix}, \quad \boldsymbol{\Sigma}_k = \begin{bmatrix} \boldsymbol{\Sigma}_k^{(i)} & \boldsymbol{\Sigma}_k^{(i,o)} \\ \boldsymbol{\Sigma}_k^{(o,i)} & \boldsymbol{\Sigma}_k^{(o)} \end{bmatrix} \tag{4.27}$$

Gaussian Mixture Regression relies on the joint probability distribution $p(\boldsymbol{\xi}_j^{(i)}, \boldsymbol{\xi}_j^{(o)})$ of the GMM to infer the the conditional distribution of the output vector given the input vector $p(\boldsymbol{\xi}_j^{(o)} \mid \boldsymbol{\xi}_j^{(i)})$. The resulting multimodal distribution can be approximated by a single Gaussian

$$p(\boldsymbol{\xi}_j^{(o)} \mid \boldsymbol{\xi}_j^{(i)}) \sim \mathcal{N}\left(\widetilde{\boldsymbol{\mu}}_j, \widetilde{\boldsymbol{\Sigma}}_j\right) \tag{4.28}$$

$$\widetilde{\boldsymbol{\mu}}_j = \sum_{k=1}^{K} h_k(\boldsymbol{\xi}_j^{(i)}) \widetilde{\boldsymbol{\mu}}_k(\boldsymbol{\xi}_j^{(i)}) \tag{4.29}$$

$$\widetilde{\boldsymbol{\Sigma}}_j = \sum_{k=1}^{K} h_k(\boldsymbol{\xi}_j^{(i)}) \left(\widetilde{\boldsymbol{\Sigma}}_k + \widetilde{\boldsymbol{\mu}}_k(\boldsymbol{\xi}_j^{(i)}) \widetilde{\boldsymbol{\mu}}_k(\boldsymbol{\xi}_j^{(i)})^T\right) - \widetilde{\boldsymbol{\mu}}_j \widetilde{\boldsymbol{\mu}}_j^T \tag{4.30}$$

with componentwise conditional means and covariances

$$\widetilde{\boldsymbol{\mu}}_k(\boldsymbol{\xi}_j^{(i)}) = \boldsymbol{\mu}_k^{(o)} + \boldsymbol{\Sigma}_k^{(o,i)} \left(\boldsymbol{\Sigma}_k^{(i)}\right)^{-1} \left(\boldsymbol{\xi}_j^{(i)} - \boldsymbol{\mu}_j^{(i)}\right) \tag{4.31}$$

$$\widetilde{\boldsymbol{\Sigma}}_k = \boldsymbol{\Sigma}_k^{(o)} - \boldsymbol{\Sigma}_k^{(o,i)} \left(\boldsymbol{\Sigma}_k^{(i)}\right)^{-1} \boldsymbol{\Sigma}_k^{(i,o)} \tag{4.32}$$

Finally, the responsability $h_k(\boldsymbol{\xi}_j^{(i)})$ of component k is computed in closed form as

$$h_k(\boldsymbol{\xi}_j^{(i)}) = \frac{\pi_k \cdot \mathcal{N}\left(\boldsymbol{\xi}_j^{(i)} \mid \boldsymbol{\mu}_j^{(i)}, \boldsymbol{\Sigma}_j^{(i)}\right)}{\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}\left(\boldsymbol{\xi}_j^{(i)} \mid \boldsymbol{\mu}_k^{(i)}, \boldsymbol{\Sigma}_k^{(i)}\right)} \tag{4.33}$$

In Figure 4.7, we show an example of regression with GMR for a GMM with $K = 2$. As a probabilistic approach, we can see that GMMs encode the variability of the demonstrated movements in the covariance matrices. This allows to identify those phases in the task that need to be reproduced exactly as the human teacher, or on the contrary, some deviation is admissible.

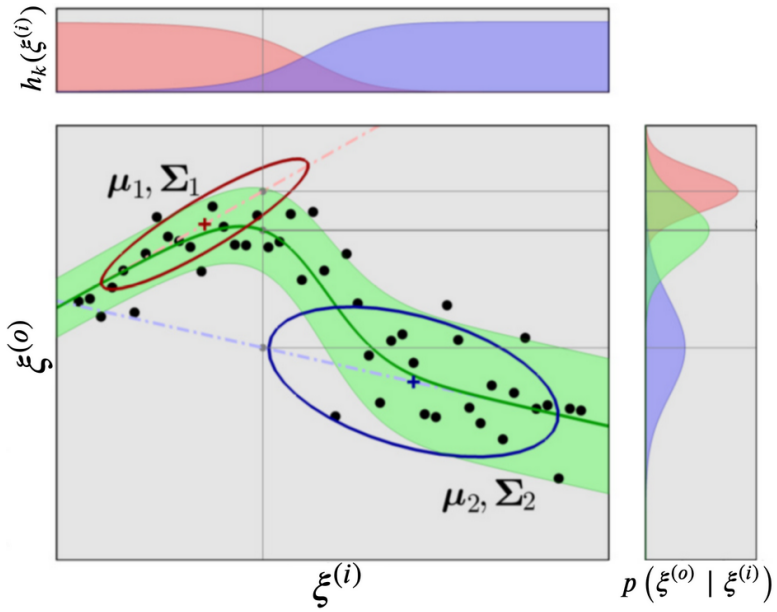

Figure 4.7: Illustration of a GMM and the corresponding GMR in two dimensions. Two Gaussian components are fitted to the demonstration data, represented by black dots. Each component describes a linear trend, whose influence on the learned trajectory is determined by the responsibility of each component (top). The conditional distribution for three different inputs is shown on the right panel (after [35]).

4.2.2 Extensions

Task-parameterized models of movements refer to representations that can automatically adapt to a set of task parameters that can, for instance, describe the current context. Task-parameterized Gaussian Mixture Models (TP-GMMs) [35] aim at increasing the generalization capabilities of the learned skill by exploiting the functional nature of parameters related to different frames of reference.

Before discussing the formal aspects of the task-parameterized model, consider a set of demonstrations observed from the perspective of two different frames (Figure 4.8). Such frames are described at each time step j by $\{\boldsymbol{p}_{j,1}, \boldsymbol{R}_{j,1}\}$ and $\{\boldsymbol{p}_{j,2}, \boldsymbol{R}_{j,2}\}$, being $\boldsymbol{p}$ the position of their origins and $\boldsymbol{R}$ the orientation of their coordinate systems. The observed movement from the perspective of the first and the second frame can be represented by $\mathcal{N}\left(\boldsymbol{\mu}^{(1)}, \boldsymbol{\Sigma}^{(1)}\right)$ and $\mathcal{N}\left(\boldsymbol{\mu}^{(2)}, \boldsymbol{\Sigma}^{(2)}\right)$, respectively. Then, for demonstrations observed from a different perspective, we can expect the data to lie within the distributions $\mathcal{N}\left(\hat{\boldsymbol{\xi}}_j^{(1)}, \hat{\boldsymbol{\Sigma}}_j^{(1)}\right)$ and $\mathcal{N}\left(\hat{\boldsymbol{\xi}}_j^{(2)}, \hat{\boldsymbol{\Sigma}}_j^{(2)}\right)$, with

$$\hat{\boldsymbol{\xi}}_j^{(\ell)} = \boldsymbol{R}_{j,\ell}\boldsymbol{\mu}^{(\ell)} + \boldsymbol{p}_{j,\ell}, \qquad \hat{\boldsymbol{\Sigma}}_j = \boldsymbol{R}_{j,\ell}\boldsymbol{\Sigma}^{(\ell)}\boldsymbol{R}_{j,\ell}^T \qquad \ell = 1, 2 \tag{4.34}$$

computed using the linear transformation properties of Gaussians.

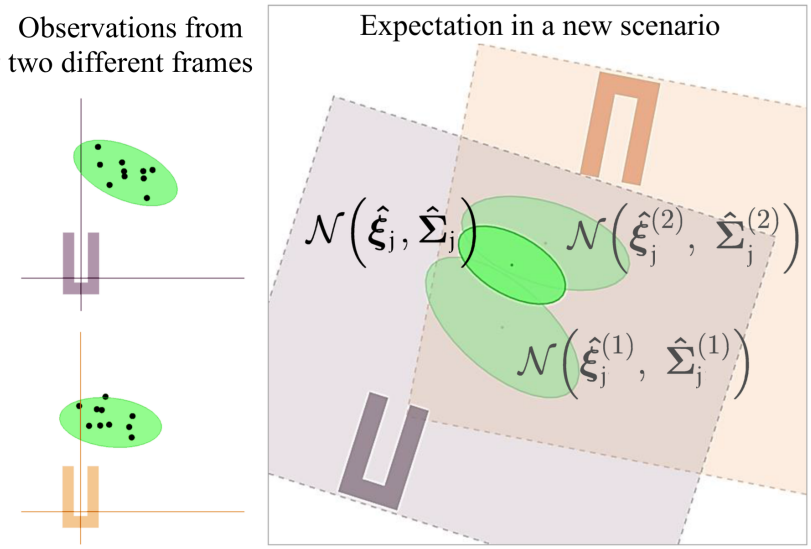

Figure 4.8: Expectation for the location of new data after observing demonstrations from two different frame observers. The estimate is obtained by minimizing equation 4.35, whose results corresponds to the product of two Gaussians (after [35]).

During the reproduction of the task trajectory, we need to determine a trade-off to concord with the distributions expected by each frame. We can formulate this as the following optimization problem

$$\hat{\boldsymbol{\xi}}_j = \arg\min_{\boldsymbol{\xi}_j} = \sum_{\ell=1}^{2} \left(\boldsymbol{\xi}_j - \hat{\boldsymbol{\xi}}_j^{(j)}\right)^T \left(\hat{\boldsymbol{\Sigma}}_j^{(j)}\right)^{-1} \left(\boldsymbol{\xi}_j - \hat{\boldsymbol{\xi}}_j^{(j)}\right) \tag{4.35}$$

Differentiating (4.35) and equating to zero, yields a point estimate $\hat{\boldsymbol{\xi}}_j$, with an uncertainty defined by a covariance $\hat{\boldsymbol{\Sigma}}_j$. As it can be intuited from Figure 4.8, the resulting distribution $\mathcal{N}\left(\hat{\boldsymbol{\xi}}_j, \hat{\boldsymbol{\Sigma}}_j\right)$ is equivalent to the product of the two Gaussians, $\mathcal{N}\left(\hat{\boldsymbol{\xi}}_j^{(1)}, \hat{\boldsymbol{\Sigma}}_j^{(1)}\right)$ and $\mathcal{N}\left(\hat{\boldsymbol{\xi}}_j^{(2)}, \hat{\boldsymbol{\Sigma}}_j^{(2)}\right)$.

Task-Parameterized Gaussian Mixture Models are a generalization of this concept to more complex movements, encoded in a GMM instead of a single Gaussian. By encoding the movements in multiple coordinate systems simultaneously, the idea is to use this information during the reproduction of the task to adapt the demonstrated skill to unforeseen scenarios.

Formally, let us assume that the task parameters are represented by L coordinate systems $\{\boldsymbol{p}_{j,\ell}, \boldsymbol{R}_{j,\ell}\}_{\ell=1}^{L}$. The demonstrations $\boldsymbol{\xi}$ are observed from these different perspectives, forming L trajectory samples $\boldsymbol{X}^{(\ell)}$

$$\boldsymbol{X}_j^{(\ell)} = \boldsymbol{R}_{j,\ell}^{-1} \left(\boldsymbol{\xi}_j - \boldsymbol{p}_{j,\ell}\right) \tag{4.36}$$

The parameters of a TP-GMM with K components are then defined by

$$\left\{\pi_k, \left\{\boldsymbol{\mu}_k^{(\ell)}, \boldsymbol{\Sigma}_k^{(\ell)}\right\}_{\ell=1}^{L}\right\}_{k=1}^{K} \tag{4.37}$$

where $\boldsymbol{\mu}_k^{(\ell)}$ and $\boldsymbol{\Sigma}_k^{(\ell)}$ are the mean and covariance matrix of the k-th Gaussian component in the candidate frame ℓ. As in Section 4.2.1, these can be determined iteratively until convergence with an expectation-maximization algorithm:

1. **E-step**: Evaluate the responsibilities $\gamma_{j,k}$, considering all candidate frames:

$$\gamma_{j,k} = \frac{\pi_k \cdot \prod_{\ell=1}^{L} \mathcal{N}\left(\boldsymbol{X}_j^{(\ell)} \mid \boldsymbol{\mu}_k^{(\ell)}, \boldsymbol{\Sigma}_k^{(\ell)}\right)}{\sum_{k=1}^{K} \pi_k \cdot \prod_{\ell=1}^{L} \mathcal{N}\left(\boldsymbol{X}_j^{(\ell)} \mid \boldsymbol{\mu}_k^{(\ell)}, \boldsymbol{\Sigma}_k^{(\ell)}\right)} \tag{4.38}$$

2. **M-step**: Update the estimate of the parameters

$$\boldsymbol{\mu}_k^{(\ell)} = \frac{1}{\sum_{j=1}^{n} \gamma_{j,k}} \sum_{j=1}^{n} \gamma_{j,k} \cdot \boldsymbol{X}_j^{(\ell)} \tag{4.39}$$

$$\boldsymbol{\Sigma}_k^{(\ell)} = \frac{1}{\sum_{j=1}^{n} \gamma_{j,k}} \sum_{j=1}^{n} \gamma_{j,k} \left(\boldsymbol{X}_j^{(\ell)} - \boldsymbol{\mu}_k^{(\ell)}\right) \left(\boldsymbol{X}_j^{(\ell)} - \boldsymbol{\mu}_k^{(\ell)}\right)^T \tag{4.40}$$

$$\pi_k = \frac{1}{n} \sum_{j=1}^{n} \gamma_{j,k} \tag{4.41}$$

These operations aim to maximize the log-likelihood but subject to the constraint that the data in the different frames arose from the same source.

The learned model can be then used to reproduce movements considering new candidate frames. A new GMM with parameters $\{\pi_k, \hat{\boldsymbol{\mu}}_{j,k}, \hat{\boldsymbol{\Sigma}}_{j,k}\}_{k=1}^{K}$ can automatically be generated with

$$\mathcal{N}\left(\hat{\boldsymbol{\mu}}_{j,k}, \hat{\boldsymbol{\Sigma}}_{j,k}\right) \propto \prod_{\ell=1}^{L} \mathcal{N}\left(\hat{\boldsymbol{\mu}}_{j,k}^{(\ell)}, \hat{\boldsymbol{\Sigma}}_{j,k}^{(\ell)}\right) \tag{4.42}$$

with

$$\hat{\boldsymbol{\mu}}_{j,k}^{(\ell)} = \boldsymbol{R}_{j,\ell} \boldsymbol{\mu}_k^{(\ell)} + \boldsymbol{p}_{j,\ell} \qquad \hat{\boldsymbol{\Sigma}}_{j,k}^{(\ell)} = \boldsymbol{R}_{j,\ell} \boldsymbol{\Sigma}_k^{(\ell)} \boldsymbol{R}_{j,\ell}^T \tag{4.43}$$

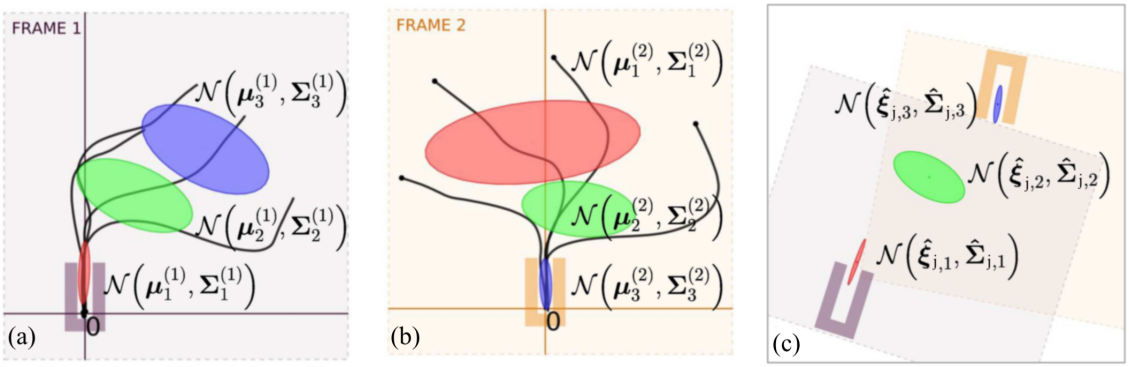

Figure 4.9: Example of a TP-GMM. (a-b) GMM encoding with $K = 3$ of the demonstrated motion from two reference frames. (c) GMM retrieved at time step j for a new observer, computed as the product of Gaussian distributions (after [35]).

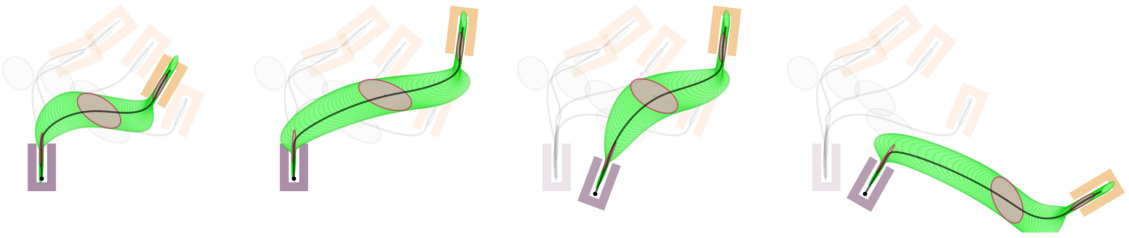

Figure 4.10: Generalization capability of a TP-GMM in combination with GMR. Each column shows a different scenario. The demonstrations and the corresponding adapted model parameters are depicted in semi-transparent colors (after [35]).

The result of the Gaussian product given by

$$\hat{\boldsymbol{\Sigma}}_{j,k} = \left(\sum_{\ell=1}^{L} \hat{\boldsymbol{\Sigma}}_{j,k}^{(\ell)^{-1}} \right)^{-1} \qquad \hat{\boldsymbol{\mu}}_{j,k} = \hat{\boldsymbol{\Sigma}}_{j,k} \sum_{\ell=1}^{L} \hat{\boldsymbol{\Sigma}}_{j,k}^{(\ell)^{-1}} \hat{\boldsymbol{\mu}}_{j,k}^{(\ell)} \tag{4.44}$$

In Figure 4.9, we depict the resulting TP-GMM for the two frame example in Figure 4.8. We can see how the variability encoded in frame 1 is low at the start of the movement and high towards frame 2, and vice-versa. Note how, by using the Gaussian product, the invariant motions are encoded with small covariances.

Finally, the GMM obtained from equation (4.44) can be used to retrieve a reference trajectory for the robot through GMR at any given time step j. In Figure 4.10, we show an illustrative example. We can observe that, by using a task-parametrized model, suitable trajectories for different unforeseen scenarios can be retrieved.

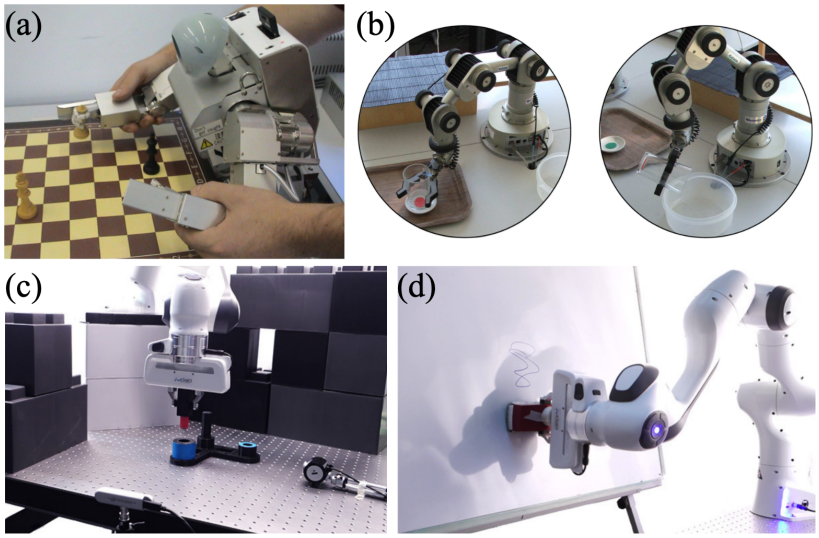

Figure 4.11: Applications of GMMs. (a) Moving a chess piece [32]. (b) Empty a glass in a basin [39] (c-d) Peg-in-hole and wiping board task, respectively [40].

4.2.3 Applications

The combination of GMM and GMR is an example of a probabilistic approach that has a proven record of success in LfD. Advantages of GMMs over DMPs include the automatic inference of the correlations between the movement variables, and the encoding of the underlying variability in the demonstrated motions.

The information encoded by the covariance matrices provides a continuous representation of the task constraints. In [32], the authors exploit this to decompose, generalize and reconstruct robot gestures. After temporally aligning the demonstrations with the Dynamic Time Warping (DTW) algorithm [41], they model the gesture required to move a chess piece with GMMs (Figure 4.11a), which allows a localized characterization of the different parts of the gesture.

The covariance matrices can also be used to quantify the relevance of a task constraint. The authors [39] present an approach based on GMMs for designing a robot controller that takes into account the task constraints in the joint and Cartesian space simultaneously by retrieving a trade-off motion. The method is illustrated by teaching a robot to empty a glass of water in a basin (Figure 4.11b).

A major challenge in LfD is handling disturbances, which might come, for instance, from modeling errors or external forces. These can lead the robot far from the states visited while learning, which in real systems might cause dangerous situations. In [40], they take advantage of GMMs to quantify the action uncertainty at each state and fuse several controllers to ensure safe execution. The proposed approach is validated through the peg-in-hole and wiping board tasks (Figure 4.11c-d).

4.3 Probabilistic Movement Primitives

Movement Primitives (MPs), as we discuss in Section 4.1, are a well-established approach for representing movement policies in robotics. In this regard, Probabilistic Movement Primitives (ProMPs) [42] are aimed to be a general probabilistic framework for representing and learning MPs.

Analogously to GMMs, a ProMP represents a distribution over trajectories. In this case, instead of a mixture of Gaussians, the probabilistic model for representing such distribution is based on basis functions. This allows reducing significantly the number of model parameters, facilitating the learning process. Working with distributions enables to formulate the desirable properties for MPs, such as generalization to new situations, or to be easy to learn from demonstrations, in terms of operations from probability theory.

Probabilistic Movement Primitives are one of the most well-established approaches in LfD because they unify many useful features from MPs in a single probabilistic framework. The method is quite recent, and it can be attributed to Alexandros Paraschos et al. Therefore, we refer the reader interested in a complete and comprehensive development to [43, 44], from which this section has been adapted.

We start, in Section 4.3.1, introducing the basic formulation for using Probabilistic Motion Primitives to learn a skill representation from demonstrated trajectories. Afterward, in Section 4.3.2, we present some probabilistic operators that can be used to further enhance the capabilities of ProMPs. In the end, in Section 4.3.3, we review some applications where this method has been adopted with success.

4.3.1 Basic Formulation

We start by considering a robot with d degrees-of-freedom, being the joint angles given by the vector $\boldsymbol{q} \in \mathbb{R}^d$. In ProMPs, a single movement execution is modelled as a trajectory $\mathcal{T} = \{\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t\}_{t=1}^T$, defined by the joint position $\boldsymbol{q}_t$ and velocity $\dot{\boldsymbol{q}}_t$ over time. At each time step, we assume that the observation vector $\boldsymbol{y}_{i,t}$ for the i-th dimension, can be expressed in terms of a linear basis function model

$$\boldsymbol{y}_{i,t} = \begin{bmatrix} q_{i,t} \\ \dot{q}_{i,t} \end{bmatrix} = \boldsymbol{\Phi}_t \boldsymbol{w}_i + \boldsymbol{\varepsilon}_y \tag{4.45}$$

where $\boldsymbol{\Phi}_t = [\boldsymbol{\phi}_t, \dot{\boldsymbol{\phi}}_t]^T$ represents the time-dependent basis function matrix for the position $q_{i,t}$ and velocity $\dot{q}_{i,t}$, $\boldsymbol{w}_i$ the weight vector, and $\boldsymbol{\varepsilon}_y \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_y)$ zero-mean independent and identically distributed Gaussian noise.

Considering the model in equation (4.45), the probability of an observation $\boldsymbol{y}_t$ at time t, given the combined weight vector $\boldsymbol{w} = [\boldsymbol{w}_1^T, \ldots, \boldsymbol{w}_d^T]^T$ is given by

$$p(\boldsymbol{y}_t \mid \boldsymbol{w}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{y}_{1,t} \\ \vdots \\ \boldsymbol{y}_{d,t} \end{bmatrix} \middle| \begin{bmatrix} \boldsymbol{\Phi}_t & \ldots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \ldots & \boldsymbol{\Phi}_t \end{bmatrix} \boldsymbol{w}, \boldsymbol{\Sigma}_y \right) = \mathcal{N}\left(\boldsymbol{y}_t \mid \boldsymbol{\Psi}_t \boldsymbol{w}, \boldsymbol{\Sigma}_y\right) \quad (4.46)$$

Then, it follows that the probability of observing a trajectory $\mathcal{T}$ is

$$p(\mathcal{T} \mid \boldsymbol{w}) = \prod_{t=1}^{T} \mathcal{N}\left(\boldsymbol{y}_t \mid \boldsymbol{\Psi}_t \boldsymbol{w}, \boldsymbol{\Sigma}_y\right) \quad (4.47)$$

Note that the covariance matrix of the distribution in (4.46) only takes into account the observation noise. In order to capture the variability of the trajectories, ProMPs introduce a distribution over the weight vector $p(\boldsymbol{w}, \boldsymbol{\theta})$, with parameters $\boldsymbol{\theta}$. Marginalizing over the weight vector equation (4.46), and assuming a Gaussian distribution for $p(\boldsymbol{w}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$ the distribution for $\boldsymbol{y}_t$ yields

$$p(\boldsymbol{y}_t, \boldsymbol{\theta}) = \int \mathcal{N}\left(\boldsymbol{y}_t \mid \boldsymbol{\Psi}_t \boldsymbol{w}, \boldsymbol{\Sigma}_y\right) \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w) d\boldsymbol{w} \quad (4.48)$$

The model parameters are then given by the observation noise variance $\boldsymbol{\Sigma}_y$ and $\boldsymbol{\theta}$. The former depends on the sensors used to gather the data, so the question now is, how can we learn $\boldsymbol{\theta} = \{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\}$ from the demonstration data? The learning procedure can be carried out by means of a simple maximum likelihood estimation algorithm. First, the weights for each individual trajectory in the demonstration set are estimated with ridge regression (Section 2.4.1)

$$\boldsymbol{w}_j = \left(\boldsymbol{\Psi}^T \boldsymbol{\Psi} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{\Psi}^T \boldsymbol{Y}_j \quad (4.49)$$

where $\boldsymbol{Y}_j$ stands for the positions and velocities of all joints and time steps from the j-th demonstration, and $\boldsymbol{\Psi}$ the corresponding basis function matrix for all t. In a second and final step, the mean $\boldsymbol{\mu}_w$ and covariance $\boldsymbol{\Sigma}_w$ are obtained from the resulting weight samples for the N demonstrations

$$\boldsymbol{\mu}_w = \frac{1}{N} \sum_{j=1}^{N} \boldsymbol{w}_j \qquad \boldsymbol{\Sigma}_w = \frac{1}{N-1} \sum_{j=1}^{N} \left(\boldsymbol{w}_j - \boldsymbol{\mu}_w\right) \left(\boldsymbol{w}_j - \boldsymbol{\mu}_w\right)^T \quad (4.50)$$

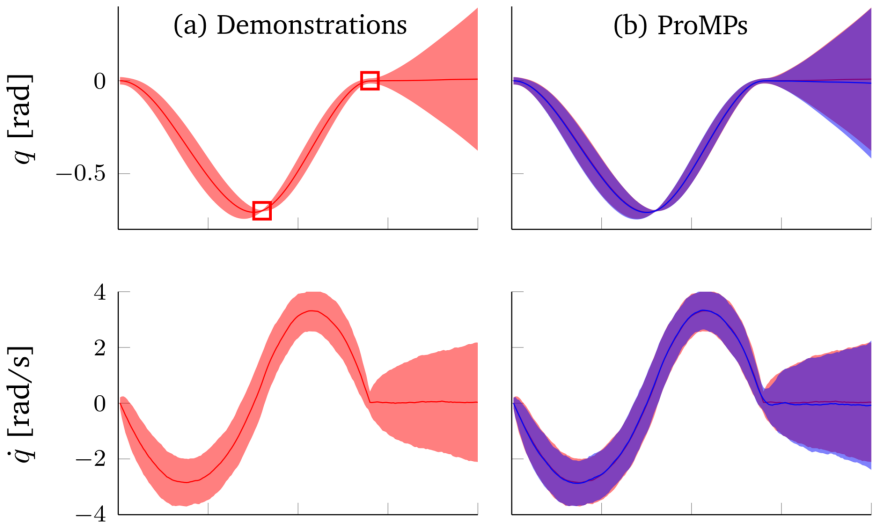

Figure 4.12: Illustrative example of ProMPs. The shaded areas represent two times the standard deviation. (a) Distribution of demonstrated trajectories, generated for a via-point task using a stochastic optimal control algorithm. (b) Learned trajectory distribution using ProMPs (blue) superposed to the demonstrations (after [43]).

In Figure 4.12, we show an example of a ProMP. The demonstrations are first generated using an optimal control algorithm [45]. Then, the trajectory distribution is learned using equations (4.49) and (4.50). We can see that the ProMP policy is capable of reproducing almost exactly the variability of the movement.

4.3.2 Extensions

Taking advantage of the probabilistic nature of ProMPs, probability operators can be used to further enhance their capabilities. In particular, we discuss how to modulate the trajectory distribution by conditioning, and combine and blend ProMPs.

Modulation with Via-Points

The modulation of the learned trajectory distribution using via-points is an important property to adapt the movement to new scenarios. In probabilistic terms, this can be achieved by conditioning the ProMP to reach a certain state $\boldsymbol{y}_t^v$ at time t. Adding the observation $\mathcal{V} = \{\boldsymbol{y}_t^v, \boldsymbol{\Sigma}_y^v\}$, and applying the Bayes' theorem yields

$$p(\boldsymbol{w} \mid \mathcal{V}) \propto \mathcal{N}\left(\boldsymbol{y}_t^v \mid \boldsymbol{\Psi}_t \boldsymbol{w}, \boldsymbol{\Sigma}_y^v\right) \mathcal{N}\left(\boldsymbol{w} \mid \boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\right) \tag{4.51}$$

where $\boldsymbol{\Sigma}_y^v$ describes the allowed deviation of the resulting motion from the via-point.

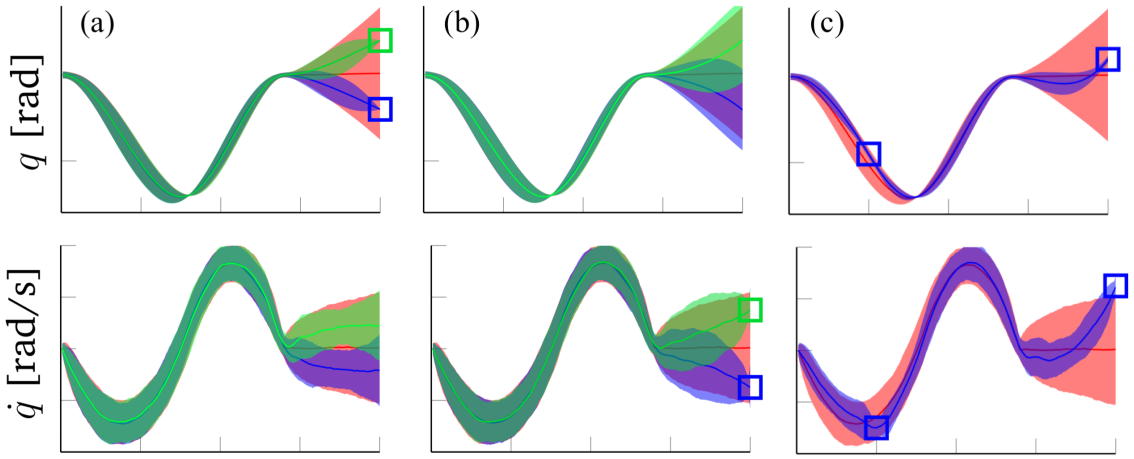

Figure 4.13: Modulation of ProMPs using via-points, depicted as blue and green boxes. (a) Via-points on the final position. (b) Via-points on the final velocity. (c) Via-points in intermediate and final locations for the position and velocity simultaneously (after [43]).

The conditional distribution $p(\boldsymbol{w} \mid \mathcal{V})$ is again Gaussian with mean and variance

$$\boldsymbol{\mu}_w^v = \boldsymbol{\mu}_w + \boldsymbol{\Sigma}_w \boldsymbol{\Psi}_t \left(\boldsymbol{\Sigma}_y^v + \boldsymbol{\Psi}_t^T \boldsymbol{\Sigma}_w \boldsymbol{\Psi}_t\right)^{-1} \left(\boldsymbol{y}_t^v - \boldsymbol{\Psi}_t^T \boldsymbol{\mu}_w\right) \tag{4.52}$$

$$\boldsymbol{\Sigma}_w^v = \boldsymbol{\Sigma}_w - \boldsymbol{\Sigma}_w \boldsymbol{\Psi}_t \left(\boldsymbol{\Sigma}_y^v + \boldsymbol{\Psi}_t^T \boldsymbol{\Sigma}_w \boldsymbol{\Psi}_t\right)^{-1} \boldsymbol{\Psi}_t^T \boldsymbol{\Sigma}_w \tag{4.53}$$

An illustration of a ProMP conditioned to different target positions and/or velocities is shown in Figure 4.13. We can see that the trajectory distribution adapts successfully, staying within the original demonstrations far from the via-points, while also reaching the specified states at the corresponding time steps.

Combination and Blending of ProMPs

ProMPs take advantage of the product of trajectory distributions to continuously combine or blend several of them into a single movement. Assuming that we have a set of K different primitives, these can be combined into a single one by taking the product of the individual distributions

$$p_{\text{combine}}(\mathcal{T}) \propto \prod_{k=1}^{K} p_k(\mathcal{T}_k)^{\alpha^{(k)}} \tag{4.54}$$

where the factor $\alpha^{(k)} \in [0, 1]$ denotes the activation of the k-th primitive. The resulting product is a trade-off between the combined ProMPs, capturing the overlapping region of the trajectory space where all have a high probability mass.

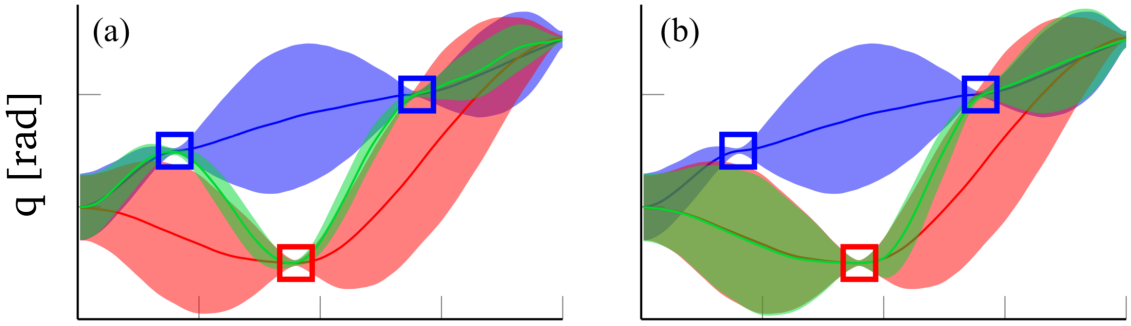

Figure 4.14: Combination (a) and blending (b) of two ProMPs. The resulting distribution is represented in green color, and the original ProMPs in blue and red (after [43]).

Alternatively, instead of activating all the ProMPs simultaneously, we might want to modulate the activation in order to continuously blend the movement from one primitive to the next one. For achieving this, we can use time-varying activation functions $\alpha_t^{(k)}$

$$p_{\text{blend}}(\mathcal{T}) \propto \prod_{t=1}^{T} p_{\text{blend}}(\boldsymbol{y}_t) = \prod_{t=1}^{T} \prod_{k=1}^{K} \mathcal{N}\left(\boldsymbol{y}_t^{(k)} \mid \boldsymbol{\mu}_t^{(k)}, \boldsymbol{\Sigma}_t^{(k)}\right)^{\alpha_t^{(k)}} \tag{4.55}$$

The resulting distribution at each time step $p_{\text{blend}}(\boldsymbol{y}_t)$ is Gaussian, with mean $\boldsymbol{\mu}_{t,\text{blend}}$ and covariance $\boldsymbol{\Sigma}_{t,\text{blend}}$ given by

$$\boldsymbol{\mu}_{t,\text{blend}} = \boldsymbol{\Sigma}_{t,\text{blend}} \left(\sum_{k=1}^{K} \left(\boldsymbol{\Sigma}_t^{(k)} \,/\, \alpha_t^{(k)} \right)^{-1} \boldsymbol{\mu}_t^{(k)} \right) \tag{4.56}$$

$$\boldsymbol{\Sigma}_{t,\text{blend}} = \left(\sum_{k=1}^{K} \left(\boldsymbol{\Sigma}_t^{(k)} \,/\, \alpha_t^{(k)} \right)^{-1} \right)^{-1} \tag{4.57}$$

An example of the combination and blending of Probabilistic Movement Primitives is depicted in Figure 4.14. Two different ProMPs are learned for solving the defined indicated by the via-points represented with the same color. On the left-hand side, we can see that the combined ProMP reaches all three via-point simultaneously i.e. both tasks are performed successfully at the same time. On the right-hand side, the blended movement first follows the red ProMP and, subsequently, switches to following exactly the blue ProMP.

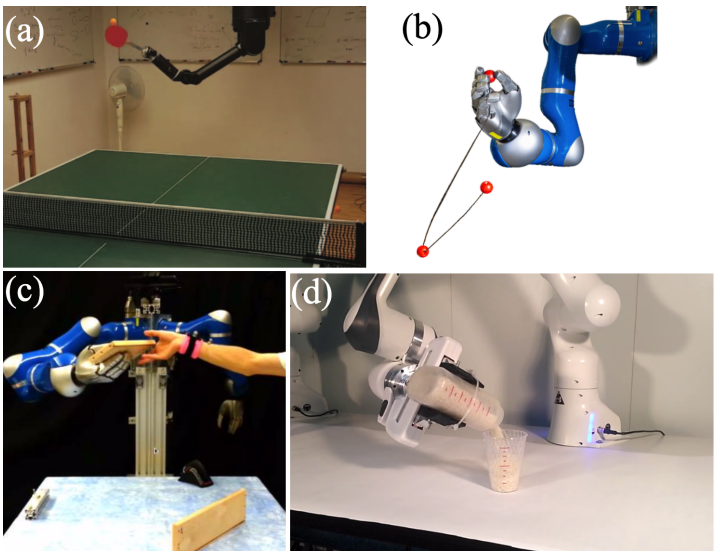

Figure 4.15: Applications of ProMPs. (a) Play table tennis (after [46]). (b) Play Astrojax (after [43]). (c) Collaborative toolbox assembly (after [47]). (d) Pouring task (after [48]).

4.3.3 Applications

Probabilistic Movement Primitives have been successfully used for learning different robotic tasks from demonstrations. The main advantages of ProMPs lie in their capability to generalize the learned task by conditioning the trajectory distribution to some desired keypoints and their efficient learning procedure.

Due to its strong requirements in terms of human-like motion capabilities, robot table tennis is an interesting test bench for robot learning approaches (Figure 4.15a). In [46], the authors use ProMPs to learn table tennis strokes. They exploit the capabilities to adapt the motion primitive to successfully strike and return the ball. ProMPs have also been used for learning rythmic motions such as playing the game Astrojax [43] (Figure 4.15b). By capturing the variability in the demonstrations, the robot is capable of generating periodic movements which show the same type of variations, allowing it to successfully perform the task.

In physical human-robot interaction, tasks are challenging due to the inherent variability of humans. This uncertainty during the interaction motivates in [47] the use of ProMPs. The authors use them for both, learning the human intent, and the corresponding robot commands, validating their approach through the collaborative assembly of a toolbox (Figure 4.15c). Another central problem in LfD is that the acquisition of demonstrations is sometimes costly. Active learning is a promising research direction since it allows the robot to actively request the demonstration, reducing the human burden of choosing which to provide. In [48], the authors address this problem using ProMPs due to their efficient learning procedure, demonstrating the effectiveness of their method on a pouring task (Figure 4.15d).

4.4 Kernelized Movement Primitives

In the context of learning from demonstration, DMPs, GMMs, and ProMPs (Sections 4.1, 4.2 and 4.3, respectively) have achieved reliable performance. However, they still have some limitations. The main weaknesses of DMPs are first, their deterministic nature, which does not allow to encode the variability in the demonstrations; and second, the use of manually defined basis functions, which limit their applicability for high-dimensional learning problems. The latter is also applicable to ProMPs. GMMs on the other hand, alleviate the modeling of trajectories via specific functions. However, the adaptation to via-points requires re-estimate the entire model. Kernelized Movement Primitives (KMPs) [49] aims to overcome the aforementioned limitations. The comparison between the most relevant features of the methods discussed throughout the chapter are summarized in Table 4.1

Feature	DMPs	GMMs	ProMPs	KMPs
Probabilistic	—	✓	✓	✓
Via-points	✓	—	✓	✓
High-dimensional learning	—	✓	—	✓

Table 4.1: Comparsion between DMPs, GMMs, ProMPs and KMPs.

Kernelized Movement Primitives provide a non-parametric solution for learning a distribution of demonstrations, alleviating the use of basis functions. By taking advantage of the kernel treatment, their capability of learning demonstrations associated with high-dimensional inputs is further improved. Additionally, the probabilistic representation of KMPs allows them to exploit operators from probability theory to adapt the learned motion.

KMPs are a very recent method, and for this reason, the related literature is not very extensive. It can be attributed to Yanlong Huang et al., being [49] the reference paper, and which the contents of this chapter are based on. For those readers interested in the implementation details, we also refer them to the KMPs software repository [50], also published by the authors of the method.

We start, in Section 4.4.1, introducing the fundamental formalism for learning a Kernelized Motion Primitive from demonstrations. Afterward, in Section 4.4.2, we present some extensions that allow to modulate the learned trajectory distribution and superpose several KMPs into a single one. To conclude, in Section 4.4.3, we discuss some example robotic applications for which KMPs have been adopted successfully.

4.4.1 Basic Formulation

Learning from multiple demonstrations allows for capturing the relevant features of the task. Formally, let us denote the demonstration dataset by

$$\mathcal{D} = \left\{ \{\boldsymbol{x}_{i,j}, \boldsymbol{y}_{i,j}\}_{i=1}^{N} \right\}_{j=1}^{M} \tag{4.58}$$

where $\boldsymbol{x}_{i,j}$ and $\boldsymbol{y}_{i,j}$ are the input and output variables respectively, N denotes the number of demonstrations, and M the trajectory length. We start the derivation of KMPs by considering that the taught task can be described by a parametric trajectory of the form

$$\boldsymbol{y}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{\phi}(\boldsymbol{x}) & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \boldsymbol{\phi}(\boldsymbol{x}) \end{bmatrix} \boldsymbol{w} = \boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{w} \tag{4.59}$$

where $\boldsymbol{\phi}(\boldsymbol{x})$ denotes the basis functions and $\boldsymbol{w}$ the weight vector. Assuming that $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$, the parametric trajectory is distributed according to

$$\boldsymbol{y}(\boldsymbol{x}) \sim \mathcal{N}\left(\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{\mu}_w, \boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{\Sigma}_w\boldsymbol{\Phi}(\boldsymbol{x})^T\right) \tag{4.60}$$

For an unknown $\boldsymbol{\mu}_w$ and $\boldsymbol{\Sigma}_w$, the LfD problem is equivalent to determine the values of these parameters such that (4.60) explains the observations in (4.58). To do so, KMPs optimize the parametric trajectory so that it matches the following reference distribution at each input location $\boldsymbol{x}_j$

$$p(\hat{\boldsymbol{y}}_i \mid \boldsymbol{x}_i) \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_i, \hat{\boldsymbol{\Sigma}}_i) \tag{4.61}$$

This reference trajectory distribution can be inferred directly from the demonstrations using GMMs and GMR (Section 4.2.1). Using the Kullback–Leibler (KL) divergence [51] as a measure of the distance between two probability distributions, the optimization can be formulated as the following cost function

$$J(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w) = \sum_{i=1}^{N} D_{KL}\left(\mathcal{N}\left(\boldsymbol{\Phi}(\boldsymbol{x}_i)\boldsymbol{\mu}_w, \boldsymbol{\Phi}(\boldsymbol{x}_i)\boldsymbol{\Sigma}_w\boldsymbol{\Phi}(\boldsymbol{x}_i)^T\right) \,||\, \mathcal{N}(\hat{\boldsymbol{\mu}}_i, \hat{\boldsymbol{\Sigma}}_i)\right) \tag{4.62}$$

where $D_{KL}(\cdot \,||\, \cdot)$ denotes the KL divergence between the probability distributions.

By using the properties of KL divergence between two Gaussian distributions, the optimization of (4.62) can be decomposed into a mean minimization subproblem

$$\boldsymbol{\mu}_{w*} = \arg\min_{\boldsymbol{\mu}_w} \sum_{i=1}^{N} \left(\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_i\right)^T \hat{\boldsymbol{\Sigma}}_i^{-1} \left(\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_i\right) \tag{4.63}$$

and a covariance minimization subproblem

$$\boldsymbol{\Sigma}_{w*} = \arg\min_{\boldsymbol{\Sigma}_w} \sum_{i=1}^{N} \left(-\log\left|\boldsymbol{\Phi}(\boldsymbol{x}_i)\boldsymbol{\Sigma}_w\boldsymbol{\Phi}(\boldsymbol{x}_i)^T\right| + \operatorname{tr}\left(\hat{\boldsymbol{\Sigma}}_i^{-1}\boldsymbol{\Phi}(\boldsymbol{x}_i)\boldsymbol{\Sigma}_w\boldsymbol{\Phi}(\boldsymbol{x}_i)^T\right)\right) \tag{4.64}$$

where $|\cdot|$ and $\operatorname{tr}(\cdot)$ denote the determinant and trace of a matrix, respectively.

Mean Prediction of KMPs

Note that equation (4.63) resembles a least-squares formulation, where each data point is weighted by $\hat{\boldsymbol{\Sigma}}_i^{-1}$. Thus, large deviations from the reference trajectory points with low covariance are heavily penalized. In order to circumvent overfitting, KMPs also include in 4.63 a penalty term $\lambda_\mu \boldsymbol{\mu}_w^T \boldsymbol{\mu}_w$, resulting in the optimal solution [52]

$$\boldsymbol{\mu}_{w*} = \boldsymbol{\Psi}\left(\boldsymbol{\Psi}^T\boldsymbol{\Psi} + \lambda_\mu \hat{\boldsymbol{\Sigma}}\right)^{-1} \boldsymbol{\Psi}^T \hat{\boldsymbol{\mu}} \tag{4.65}$$

where

$$\boldsymbol{\Psi}^T = \begin{bmatrix} \boldsymbol{\Phi}(\boldsymbol{x}_1)^T \\ \vdots \\ \boldsymbol{\Phi}(\boldsymbol{x}_N)^T \end{bmatrix}^T \qquad \hat{\boldsymbol{\Sigma}} = \begin{bmatrix} \hat{\boldsymbol{\Sigma}}_1 & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \hat{\boldsymbol{\Sigma}}_N \end{bmatrix} \qquad \boldsymbol{\mu} = \begin{bmatrix} \hat{\boldsymbol{\mu}}_1 \\ \vdots \\ \hat{\boldsymbol{\mu}}_N \end{bmatrix} \tag{4.66}$$

Subsequently, for a new input $\boldsymbol{x}_*$, its corresponding mean prediction $\boldsymbol{\mu}_*$ is

$$\mathbb{E}\left[\boldsymbol{y}(\boldsymbol{x}_*)\right] = \boldsymbol{\mu}_* = \boldsymbol{\Phi}(\boldsymbol{x}_*)\boldsymbol{\mu}_{w*} = \boldsymbol{\Psi}(\boldsymbol{x}_*)\boldsymbol{\Psi}\left(\boldsymbol{\Psi}^T\boldsymbol{\Psi} + \lambda_\mu \hat{\boldsymbol{\Sigma}}\right)^{-1} \boldsymbol{\Psi}^T \hat{\boldsymbol{\mu}} \tag{4.67}$$

In order to avoid the explicit definition of basis functions, equation (4.67) can be defined in terms of a kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$, which can be further rewritten as a kernel matrix $\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}')$

$$k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x}')^T \longrightarrow \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{\Phi}(\boldsymbol{x}')^T = k(\boldsymbol{x}, \boldsymbol{x}')\boldsymbol{I} \tag{4.68}$$

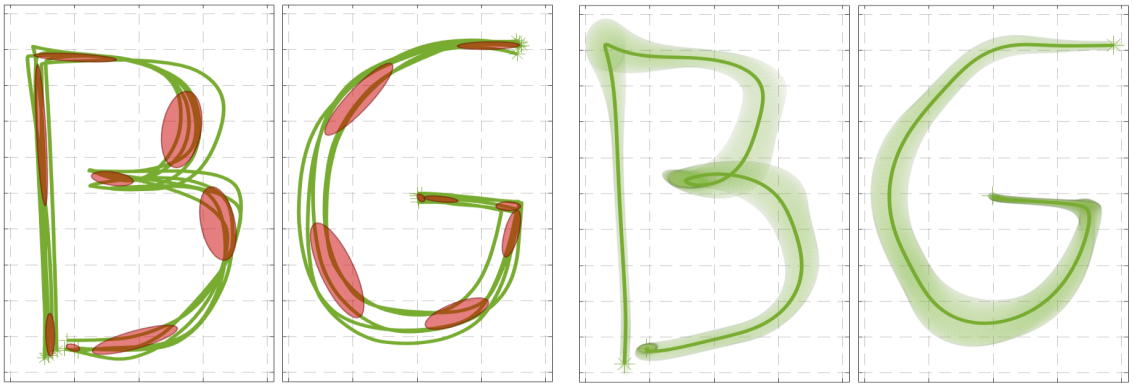

Figure 4.16: Learning of the handwritten letters 'B' and 'G' with KMPs. On the two left panels, the demonstrations are represented as green solid lines, starting from '$*$' and ending in '+'. The ellipses depict the Gaussian components of the GMMs. On the two right panels, learned trajectory distributions using KMPs, where the curve and the shaded area correspond to the mean and standard deviation, respectively (after [53]).

Hence, the mean prediction results

$$\boldsymbol{\mu}_* = \boldsymbol{K}_* \left(\boldsymbol{K} + \lambda_\mu \hat{\boldsymbol{\Sigma}} \right)^{-1} \hat{\boldsymbol{\mu}} \tag{4.69}$$

with matrices $\boldsymbol{K}$ and $\boldsymbol{K}_*$ denoting

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{k}(\boldsymbol{x}_1, \boldsymbol{x}_1) & \boldsymbol{k}(\boldsymbol{x}_1, \boldsymbol{x}_2) & \dots & \boldsymbol{k}(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \boldsymbol{k}(\boldsymbol{x}_2, \boldsymbol{x}_1) & \boldsymbol{k}(\boldsymbol{x}_2, \boldsymbol{x}_2) & \dots & \boldsymbol{k}(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{k}(\boldsymbol{x}_N, \boldsymbol{x}_1) & \boldsymbol{k}(\boldsymbol{x}_N, \boldsymbol{x}_2) & \dots & \boldsymbol{k}(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix} \quad \boldsymbol{K}_*^T = \begin{bmatrix} \boldsymbol{k}(\boldsymbol{x}_*, \boldsymbol{x}_1) \\ \boldsymbol{k}(\boldsymbol{x}_*, \boldsymbol{x}_2) \\ \vdots \\ \boldsymbol{k}(\boldsymbol{x}_*, \boldsymbol{x}_N) \end{bmatrix} \tag{4.70}$$

Covariance Prediction of KMPs

Similar to the development for the mean prediction, a penalty term $\lambda_\Sigma \text{tr}(\boldsymbol{\Sigma}_w)$ is introduced in (4.64) to bound the covariance. The covariance prediction $\boldsymbol{\Sigma}_*$ for a new input $\boldsymbol{x}_*$ results

$$\boldsymbol{\Sigma}_* = \frac{N}{\lambda_\Sigma} \left(\boldsymbol{k}(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{K}_* \left(\boldsymbol{K} + \lambda_\Sigma \hat{\boldsymbol{\Sigma}} \right)^{-1} \boldsymbol{K}_*^T \right) \tag{4.71}$$

In Figure 4.16, we show an illustrative example of a KMP. The aim is to learn to write two letters from five handwritten demonstrations each. We can clearly see that the learned trajectory distribution resembles successfully the taught skill.

4.4.2 Extensions

As we have already discussed throughout the chapter, in dynamic and unstructured environments, the capability to adapt the robot's motion is essential. Another challenge arises when the robot is given a set of candidate feasible trajectories for performing a task. These possible movements can be superposed to retrieve a motion that balances the different solutions according to their priorities. In this section, we extend the KMPs' formulation to address these two problems.

Modulation of KMPs

Kernelized Movement Primitives can be modulated by adapting the learned trajectory distribution to pass through via-points. Formally, let us define V new desired points $\mathcal{V} = \{\boldsymbol{x}_v, \boldsymbol{y}_v\}_{v=1}^{V}$, with associated conditional probability distributions

$$p(\boldsymbol{y}_v \mid \boldsymbol{x}_v) \sim \mathcal{N}\left(\boldsymbol{\mu}_v, \boldsymbol{\Sigma}_v\right) \tag{4.72}$$

These distributions can be designed based on new requirements. For instance, if the robot needs to pass through the v-th via-point with high precision, we should assign it a small covariance $\boldsymbol{\Sigma}_v$. Now, let $\widetilde{\mathcal{D}}$ and be $\widetilde{\mathcal{V}}$ the datasets corresponding to the demonstrations' and via-points' distributions, respectively

$$\widetilde{\mathcal{D}} = \left\{\boldsymbol{x}_i, \hat{\boldsymbol{\mu}}_i, \hat{\boldsymbol{\Sigma}}_i\right\}_{i=1}^{N} \qquad \widetilde{\mathcal{V}} = \left\{\boldsymbol{x}_v, \hat{\boldsymbol{\mu}}_v, \hat{\boldsymbol{\Sigma}}_v\right\}_{v=1}^{V} \tag{4.73}$$

For enforcing the KMP to pass through the desired via-points, we can simply concatenate $\widetilde{\mathcal{D}}$ and $\widetilde{\mathcal{V}}$ into an extended reference dataset. Then, we can use equations (4.69) and (4.71), taking the extended dataset instead of just $\widetilde{\mathcal{D}}$ to predict the mean and covariance. An example of the modulation of a KMP is depicted in Figure 4.17.

Superposition of KMPs

Kernelized Movement Primitives representing different feasible solutions for a task can be mixed on a single one. Formally, given a set of L reference trajectory distributions, associated with inputs and corresponding importance $\gamma_{i,\ell}$, denoted as

$$\mathcal{S} = \left\{\{\boldsymbol{x}_i, \hat{\boldsymbol{y}}_{i,\ell}, \gamma_{i,\ell}\}_{i=1}^{N}\right\}_{\ell=1}^{L} \tag{4.74}$$

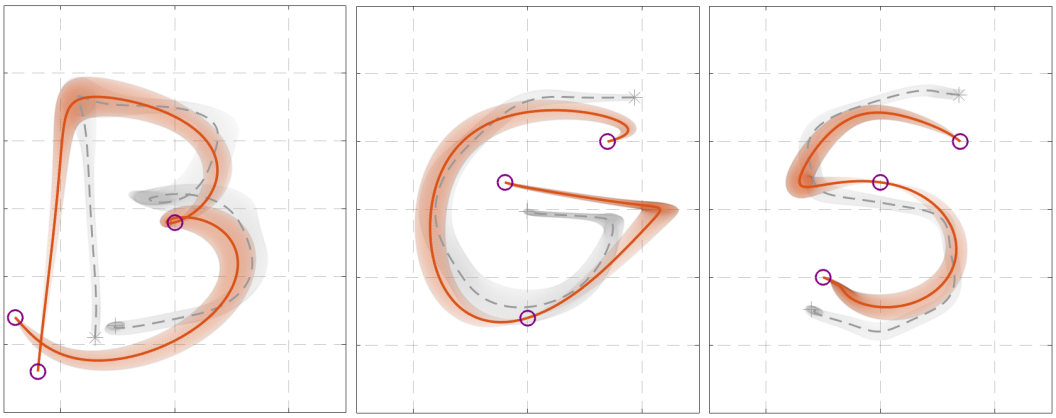

Figure 4.17: Modulation of a KMP using three via-points (purple circles): at the start, in the middle, and at the end of the motion. The gray area represents the original KMP, and the red one the modulated KMP (after [49]).

For a given input $\boldsymbol{x}_i$, the conditional trajectory distribution is given by

$$p(\hat{\boldsymbol{y}}_{i,\ell} \mid \boldsymbol{x}_i) \sim \mathcal{N}\left(\hat{\boldsymbol{\mu}}_{i,\ell}, \hat{\boldsymbol{\Sigma}}_{i,\ell}\right) \tag{4.75}$$

Additionally, the priority coefficients satisfy that $\gamma_{\ell,j} \in [0,1]$ and $\sum_{\ell=1}^{L} \gamma_{\ell,j} = 1$. To consider all the candidate trajectories along with their priority, the optimization problem in (4.62) can be modified as follows

$$J\left(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\right) = \sum_{i,\ell=1}^{N,L} \gamma_{i,\ell} D_{KL}\left(\mathcal{N}\left(\boldsymbol{\Phi}(\boldsymbol{x}_i)\boldsymbol{\mu}_w, \boldsymbol{\Phi}(\boldsymbol{x}_i)\boldsymbol{\Sigma}_w\boldsymbol{\Phi}(\boldsymbol{x}_i)^T\right) \,||\, \mathcal{N}(\hat{\boldsymbol{\mu}}_{i,\ell}, \hat{\boldsymbol{\Sigma}}_{i,\ell})\right) \tag{4.76}$$

Similar to the decomposition in (4.63) and (4.64), the objective function (4.76) can be separated into a weighted mean and covariance minimization subproblems

$$\boldsymbol{\mu}_{w*} = \arg\min_{\boldsymbol{\mu}_w} \sum_{i=1}^{N} \left(\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{\mu}_w - \bar{\boldsymbol{\mu}}_i\right)^T \bar{\boldsymbol{\Sigma}}_i^{-1} \left(\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{\mu}_w - \bar{\boldsymbol{\mu}}_i\right) \tag{4.77}$$

$$\boldsymbol{\Sigma}_{w*} = \arg\min_{\boldsymbol{\Sigma}_w} \sum_{i=1}^{N} \left(-\log\left|\boldsymbol{\Phi}(\boldsymbol{x}_i)\boldsymbol{\Sigma}_w\boldsymbol{\Phi}(\boldsymbol{x}_i)^T\right| + \operatorname{tr}\left(\bar{\boldsymbol{\Sigma}}_i^{-1}\boldsymbol{\Phi}(\boldsymbol{x}_i)\boldsymbol{\Sigma}_w\boldsymbol{\Phi}(\boldsymbol{x}_i)^T\right)\right) \tag{4.78}$$

where $\bar{\boldsymbol{\mu}}_i$ and $\bar{\boldsymbol{\Sigma}}_i$ are equal to

$$\bar{\boldsymbol{\mu}}_i = \bar{\boldsymbol{\Sigma}}_i \sum_{\ell=1}^{L} \left(\hat{\boldsymbol{\Sigma}}_{i,\ell} \,/\, \gamma_{i,\ell}\right)^{-1} \hat{\boldsymbol{\mu}}_{i,\ell} \qquad \bar{\boldsymbol{\Sigma}}_i = \sum_{\ell=1}^{L} \left(\hat{\boldsymbol{\Sigma}}_{i,\ell} \,/\, \gamma_{i,\ell}\right)^{-1} \tag{4.79}$$

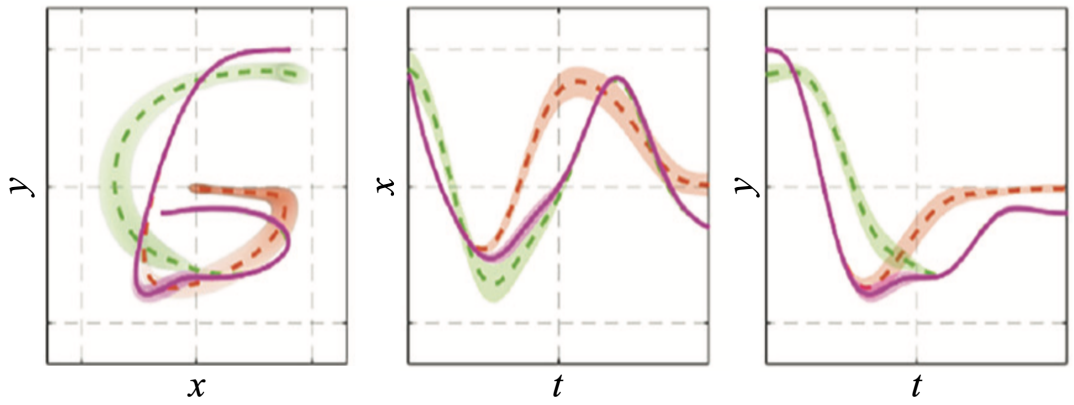

Figure 4.18: Superposition of KMPs. The resulting distribution is represented in purple, while the original ones are depicted in red and green. On the left, the $x - y$ trajectory. On the remaining right panels, the evolution over time of each coordinate (after [49]).

Note that the (4.77) and (4.78) have the same form of (4.63) and (4.64), respectively. The difference lies on the mean and covariance of the reference trajectory. Also, comparing (4.56) and (4.57) with (4.79), we can see that the new mean and covariance are equivalent to a product of Gaussians

$$\mathcal{N}\left(\bar{\boldsymbol{\mu}}_i, \bar{\boldsymbol{\Sigma}}_i\right) \propto \prod_{\ell=1}^{L} \mathcal{N}\left(\hat{\boldsymbol{\mu}}_{i,\ell}, \hat{\boldsymbol{\Sigma}}_{i,\ell} \,/\, \gamma_{i,\ell}\right) \tag{4.80}$$

Therefore, superposing KMPs can be performed by simply taking the product of the candidate trajectory distributions as the reference trajectory distribution, weighting the covariances by the inverse of the corresponding priorities. An illustrative example of the superposition of two KMPs is provided in Figure 4.18.

4.4.3 Applications

Kernelized Movement primitives have exhibited a reliable performance in several robotic tasks. One example is provided in [54]. The authors present an approach for designing an optimal controller from demonstrations by exploiting the capability of KMPs to encode the uncertainty. More specifically, the robot can be driven with high precision when the variability in the data is low, and render the robot compliant when the uncertainty is high. The performance is validated experimentally in a human-robot collaborative painting task (Figure 4.19a).

Another possible application of KMPs is obstacle avoidance [49]. This problem is interesting because it allows the robot to avoid possible collisions when executing a task and undesired circumstances occur.

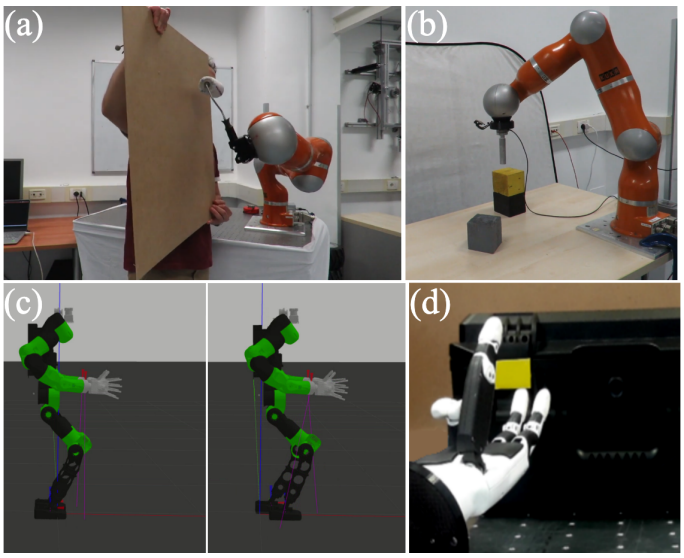

Figure 4.19: Applications of KMPs. (a) Collaborative painting (after [54]). (b) Obstacle avoidance (after [49]). (c) Walking (after [55]). (d) Unlatch a toolbox (after [56]).

Taking advantage of the adaptation of KMPs through via-points, when the robot is about to collide with an obstacle, new target locations can be defined as a function of the sensed forces (Figure 4.19b). In this way, KMPs allow generating an updated trajectory that fulfills the unforeseen constraints.

KMPs have also been used for learning walking tasks with a humanoid robot [55]. The authors use them for generating swing leg trajectories, exploiting their modulation capabilities to adjust the step location and duration. The presented approach is tested through whole-body dynamic simulations (Figure 4.19c).

As a final example, KMPs have also been used for improving the dexterity of a robot hand [56]. Due to the complexity of manually designing a controller capable of manipulating objects with dexterity, it is an ideal task to be learned from demonstrations of a human hand. The example hand postures are first encoded with KMPs, and then adapted using via-points to reuse the learned motions for unseen objects. The method is used for generating movements that allow, for instance, a robot hand to open the latches of a toolbox (Figure 4.19d).

Chapter 5

Human to Robot Motion Transfer

In robot learning from demonstration (LfD) several key problems need to be solved for ensuring such a generic approach to transferring skills across various agents and situations. These have been formulated as a set of generic questions [57]. In this chapter, we are concerned about answering *how to imitate?*. The general pipeline for human motion imitation [58], illustrated in Figure 5.2, can be divided into four stages: (1) Human motion must be observed and recorded. (2) The motion data is converted into a format suitable for imitation. (3) A model of the movement, which defines the mapping between equivalent robot and human actions is created. (4) The motion is reproduced on the robot platform. The first two stages have been addressed successfully by motion tracking systems. In the last years, these have become increasingly miniaturized and available, reducing measurement error to a minimum with advanced tracking algorithms and post-processing [59]. However, many challenges remain open regarding the correspondence between human and robot motion, and the reproduction of such complex movements.

In general, robots cannot act exactly the same way as a human does, due to differences in physical embodiment. For example, if the demonstrator uses the foot to move an object, is it acceptable for a wheeled robot to bump it, or should it use a gripper instead? To reproduce the demonstrated movement, the robot requires a mapping from the human body to its own body that allows it to reproduce the demonstrated movement. This is termed the correspondence problem [11].

The versatility of the human body when performing tasks can be partially attributed to its large number of actuated degrees of freedom (DoF). This imposes significant challenges when reproducing the human motion with a robot, due to the tight coordination that is required. On this subject, Whole-Body Control (WBC) [60] has been proposed as a promising research direction, since it opens the door for the simultaneous execution of multiple tasks, such as positioning various links while optimizing the posture, by exploiting the full capabilities of the robot's body.

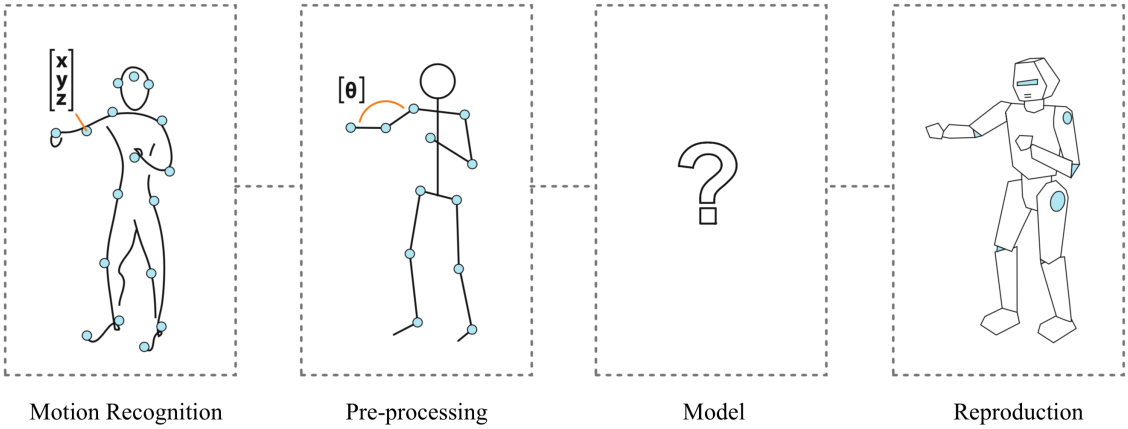

Figure 5.1: Overview of the human motion transfer pipeline. First, human motion data is measured using techniques such as motion capture. Next, motion data is processed to handle sensor noise and possibly convert the data to a more convenient format, such as joint angles. Next, a model mapping the human movement to the robot movement is formulated. Finally, the movement is reproduced on a robot (after [58]).

Another key concern in the reproduction of human motion is safety. Especially when considering robots operating in environments that are unpredictable due to the presence of people. In human environments, the robot has to feature a certain degree of compliance to be able to instantaneously react in case of contacts and physical interactions. While this can be achieved by passive elements such as mechanical springs, the concept of active compliance is dominating the field [61].

We start this chapter by addressing the correspondence problem, in Section 5.1. Beginning from its formulation, we review different solutions based on the interface used to provide demonstrations. Afterward, in Section 5.2, we look into the whole-body control paradigm. Next, in Section 5.3, we introduce a control scheme for ensuring active compliance when reproducing the human motion. Finally, in Section 5.4, we present a whole-body motion transfer framework for the TIAGo robot.

5.1 The Correspondence Problem

For learning from demonstration to be successful, the states and actions in the learning dataset must be usable by the robot [10]. Humans, and robots may perform different actions to accomplish the same task. For instance, consider the task of playing soccer. In Figure 5.2 is evident that even when performing the same task, humans and robots may interact with the environment in different ways. Humans run and kick, while robots roll and bump. Solving this discrepancy in motor capabilities implies solving the correspondence problem.

Figure 5.2: Humans and robots playing soccer. Due to differences in embodiment humans run and kick while robots roll and bump for accomplishing the same task.

The following statement of the correspondence problem, as proposed in [62], draws attention to the fact that the model and imitator agents may not necessarily share the same morphology or affordances:

Given an observed behavior of the model, which from a given starting state leads the model through a sequence of sub-goals in states, action and/or effects, one must find and execute a sequence of actions using one's own (possibly dissimilar) embodiment, which from a corresponding starting state, leads through corresponding sub-goals; in corresponding states, actions, and/or effects, while possibly responding to corresponding events.

Essentially, the idea is to find a map $f_{\mathcal{P}}^{\mathcal{R}}$ between the person's configuration space $\mathcal{P}$ and the robot's configuration space $\mathcal{R}$, such that from the human motion, the robot can reproduce the desired behavior. In order to tackle the correspondence problem systematically, we divide it into three different subproblems [63]:

1. **Observation**: In reality, we do not have access to the person state space $\mathcal{P}$. Indeed, we need an interface to observe human motion. If we define $\mathcal{O}$ as space where the observations of the motion lie, finding a solution to this problem involves the specification of the following mapping

$$f_{\mathcal{P}}^{\mathcal{O}} : \mathcal{P} \longrightarrow \mathcal{O} \tag{5.1}$$

2. **Equivalence**: Due to the differences in the embodiment, we need to determine a mapping from the observations of the demonstrator to the robot's desired behavior. This requires adequate considerations regarding the differences in the kinematic chains and joint limits. If $\mathcal{G}$ is the state space of the robot's goal configurations, the problem reduces to define

$$f_{\mathcal{O}}^{\mathcal{G}} : \mathcal{O} \longrightarrow \mathcal{G} \tag{5.2}$$

3. **Reproduction**: Generally, the desired robot behavior is not represented in terms of the robot's configuration space $\mathcal{R}$. Then, in order to be able to compute the required control actions, we need to determine the robot's configurations that allow reproducing all possible desired behaviors. This requires the computation of the following mapping

$$f_{\mathcal{R}}^{\mathcal{G}} : \mathcal{R} \longrightarrow \mathcal{G} \tag{5.3}$$

Then, the solution of the correspondence problem $f_{\mathcal{P}}^{\mathcal{R}}$ can be written using the proposed decomposition in subproblems, in terms of (5.1), (5.2) and (5.3), as

$$\begin{aligned} f_{\mathcal{P}}^{\mathcal{R}} &= f_{\mathcal{P}}^{\mathcal{O}} \circ f_{\mathcal{O}}^{\mathcal{G}} \circ \left(f_{\mathcal{R}}^{\mathcal{G}}\right)^{-1} \\ f_{\mathcal{P}}^{\mathcal{R}} &: \mathcal{P} \xrightarrow{f_{\mathcal{P}}^{\mathcal{O}}} \mathcal{O} \xrightarrow{f_{\mathcal{O}}^{\mathcal{G}}} \mathcal{G} \xrightarrow{\left(f_{\mathcal{R}}^{\mathcal{G}}\right)^{-1}} \mathcal{R} \end{aligned} \tag{5.4}$$

where $\circ$ refers to the composition operator and $()^{-1}$ to the inverse mapping. Note that the interface used to provide the demonstrations and the robot's kinematic structure play a key role in the solution of the correspondence problem. The former imposes $f_{\mathcal{P}}^{\mathcal{O}}$, while the latter restricts of $f_{\mathcal{R}}^{\mathcal{G}}$.

Since there are infinite possibilities for the robot's kinematic structure, we focus on discussing different approaches to the correspondence problem based on the selection of the interface used to provide the demonstrations. In particular, we distinguish three major trends: kinesthetic teaching (Section 5.1.1), teleoperation (Section 5.1.2), and motion capture (Section 5.1.3).

5.1.1 Kinesthetic Teaching

In kinesthetic teaching, the demonstrator holds and moves the robot along the trajectories that need to be followed to accomplish the task, while the robot actively or passively compensates for the effect of gravity [64]. Some examples are shown in Figure 5.3. This approach simplifies the correspondence problem by letting the user demonstrate the skill in the robot's environment with the robot's own capabilities. This means that $\mathcal{O} \equiv \mathcal{R}$. Changing $f_{\mathcal{O}}^{\mathcal{G}}$ by $f_{\mathcal{R}}^{\mathcal{G}}$ in equation (5.4) yields

$$f_{\mathcal{P}}^{\mathcal{R}} = f_{\mathcal{P}}^{\mathcal{O}} \circ f_{\mathcal{R}}^{\mathcal{G}} \circ \left(f_{\mathcal{R}}^{\mathcal{G}}\right)^{-1} = f_{\mathcal{P}}^{\mathcal{O}} \circ I \quad \Longrightarrow \quad f_{\mathcal{P}}^{\mathcal{R}} = f_{\mathcal{P}}^{\mathcal{O}} \tag{5.5}$$

where I is the identity function. As we can see, kinesthetic teaching directly provides the mapping between demonstrations and robot actions by projecting the observations of the human motion onto the robot's configuration space.

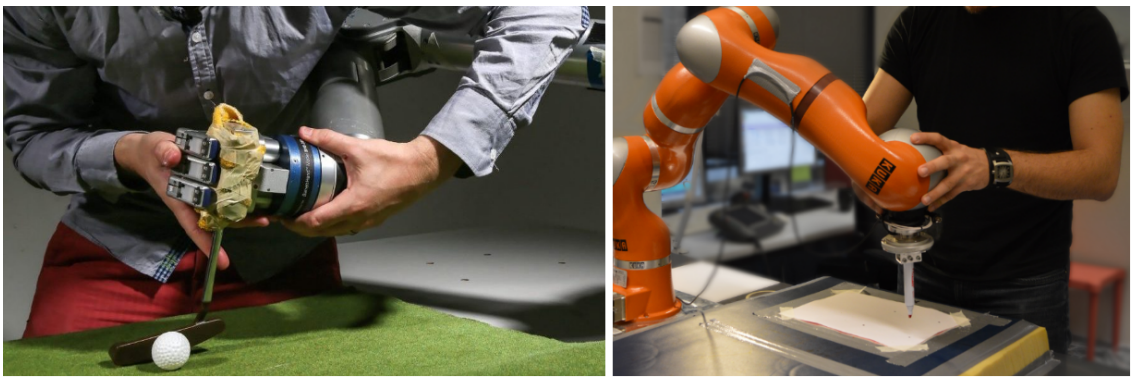

Figure 5.3: Kinesthetic teaching. The demonstrator performs the task of holding the robot.

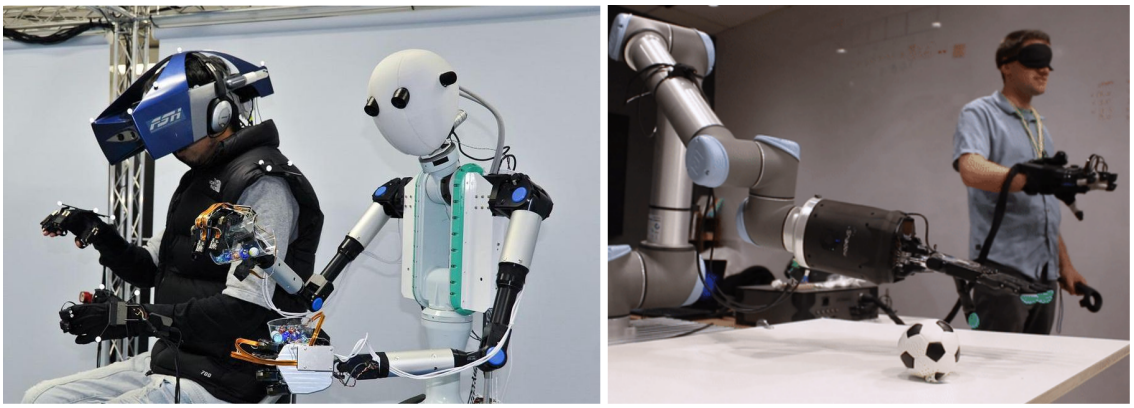

Figure 5.4: Robot teleoperation examples. On the left, Model H from Telexistence Inc. On the right, Tactile Telerobot from the Shadow Robot Company.

The main drawback of kinesthetic teaching is that the human must often use more degrees of freedom to move the robot than the number of degrees of freedom moved on the robot. This limits the type of tasks that can be taught through kinesthetic teaching. For instance, tasks that require moving both hands simultaneously cannot be taught this way.

5.1.2 Teleoperation

Using immersive teleoperation scenarios the human teacher is limited to using the robot's own sensors and effectors to perform the task. Teleoperation may be performed using simple joysticks or other remote control devices, including haptic devices. The latter has the advantage that they can allow the teacher to teach tasks that require precise control of forces, while joysticks would only provide kinematic information. Some examples are depicted in Figure 5.4.

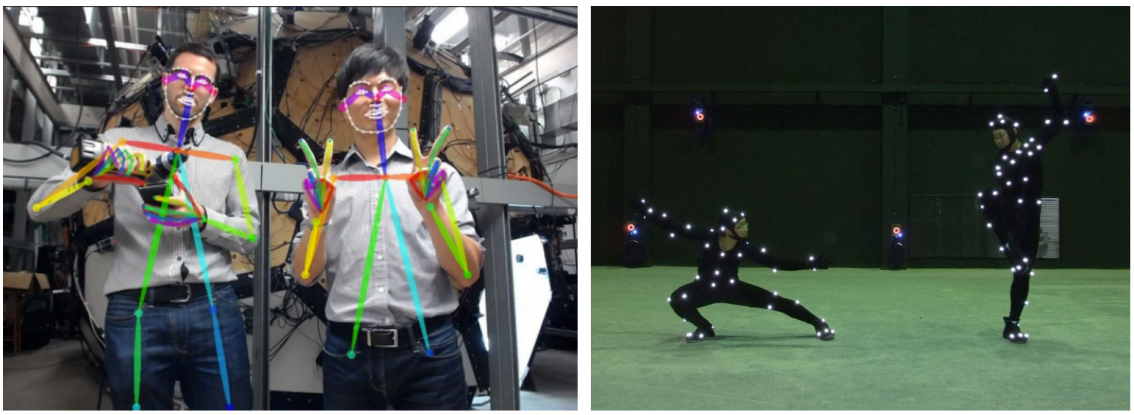

Figure 5.5: Motion capture systems. On the left, OpenPose [65] jointly detects the human body, hand, facial, and foot keypoints on single images. On the right, OptiTrack 3D tracking system. Aside from robotics, it is also very popular in video game design and animation.

Teleoperation allows solving the correspondence problem entirely since the system directly records the perception and action from the robot's configuration space $\mathcal{R}$. Similarly to kinesthetic teaching, this means that $\mathcal{O} \equiv \mathcal{R}$. Then, the correspondence between the demonstrator's actions and the robot's actions is also directly provided by the demonstration interface (5.5).

The disadvantage of teleoperation techniques is that the teacher often needs the training to learn to use the remote control device. Also, common controllers such as joysticks allow guiding only a subset of degrees of freedom. To control for all degrees of freedom, very complex, exoskeleton type of devices must be used, which can be cumbersome. This limits significantly the scope of tasks that can be taught.

5.1.3 Motion Capture

The usual approach in learning from demonstration is to directly measure the human movements using a motion capture system. Essentially, motion is recorded by tracking the precise position and orientation of points of interest at high frequency. Each tracker, usually wearable, uses fundamentally different physical principles to measure position and orientation. Mechanisms vary from using a multiplexed reading of orthogonal magnetic fields from inductive coils, accelerometers and gyroscopes, the intensity of ultrasonic pulses, the mechanical orientation of joints, or reconstruction of the position of visible markers detected with multiple cameras (Figure 5.5). The resulting dataset consists in the Cartesian trajectory of each body landmark over the duration of the motion. These methods are advantageous compared to the aforementioned ones in that they let the human move freely and perform the task naturally. This allows teaching a wide scope of tasks.

The main drawback of motion capture systems is that they require solutions to the correspondence problem. As we mentioned, trajectories are commonly recorded in Cartesian space. This means that $\mathcal{O} \equiv SE(3)^n$, where n is the number of observed body landmarks. Then, the correspondence problem can be addressed by finding a mapping between human and robot motion in Cartesian space. That is, defining $\mathcal{G} \equiv SE(3)^m$, where m is the number of robot links to be constrained. The correspondence problem solution (5.4) can then be written as

$$f_{\mathcal{P}}^{\mathcal{R}} : \mathcal{P} \xrightarrow{f_{\mathcal{P}}^{\mathcal{O}}} \mathcal{O} \equiv SE(3)^n \xrightarrow{f_{\mathcal{O}}^{\mathcal{G}}} \mathcal{G} \equiv SE(3)^m \xrightarrow{\left(f_{\mathcal{R}}^{\mathcal{G}}\right)^{-1}} \mathcal{R} \tag{5.6}$$

where $f_{\mathcal{O}}^{\mathcal{G}}$ depends on the task. The mapping $\left(f_{\mathcal{R}}^{\mathcal{G}}\right)^{-1}$ may be obtained by computing a path in Cartesian space for m robot links that are close to the path followed by the human. If the only link to be constrained is the end-effector, we can rely on standard inverse kinematics methods to find the appropriate joint displacements. For instance, the football example in Figure 5.2, would require the robot to determine a path for its center of mass which corresponds to the path followed by the human's right (or left) foot when projected on the ground. For $m \geq 2$ we must resort to more complex methods such as whole-body control (Section 5.2).

5.2 Whole-Body Control

While humans may occasionally be outperformed by robots in a single task, they are vastly more capable of adapting and combining behaviors for simultaneous execution of multiple tasks. Whole-body control (WBC) has been proposed as a promising research direction when using robots with many DoF and several simultaneous objectives, like positioning multiple links at the same time while imitating the human motion (Figure 5.6). WBC aims to: (1) define a small set of simple, low-dimensional rules; (2) that are sufficient to guarantee the correct execution of any single task, whenever feasible, and of simultaneous multiple tasks; (3) exploiting the full capabilities of the entire body of redundant robots to meet the multiple tasks' constraints [67].

There are different techniques that can be used to find a whole-body control policy [68]. In particular, we differentiate between closed-form techniques, in Section 5.2.1 and optimization-based methods 5.2.2. In the first case, the control action is derived after a sequence of algebraic operations e.g., projections, inversions, and pseudo-inversions. In the second case, the problem is defined as an optimization problem, and a solution is determined using a solver. Closed-form solutions are usually faster to compute than optimization-based solutions, but the scope of constraints that can be considered is more limited.

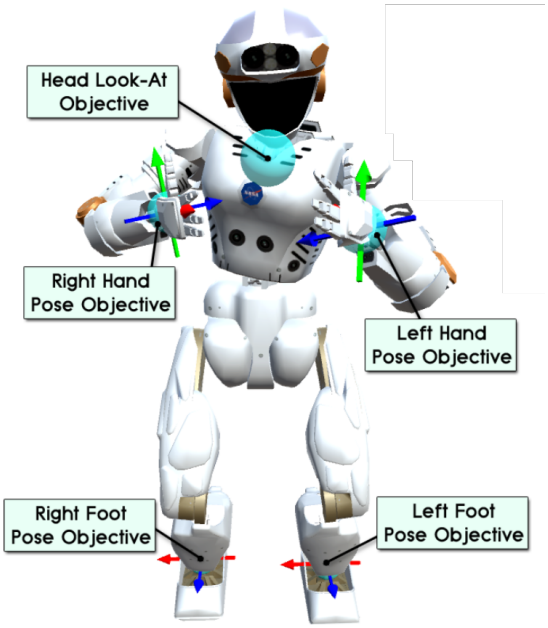

Figure 5.6: Posture control of the 61 DOF NASA Valkyrie robot involves the simultaneous execution of multiple tasks such as the positioning of both feet and hands (after [66]).

5.2.1 Closed-form WBC

Performing multiple tasks simultaneously may generate conflicting situations in which neither of the tasks is executed satisfactorily. A remedy for this inconvenience is the so-called task priority strategy, according to which a priority between the two tasks is established beforehand, and the lower priority task produces only self-motion which does not interfere with the higher priority task. In the case of a highly redundant system, such as the example depicted in Figure 5.6, it would be possible to introduce multiple constraint tasks of different nature and then decide the order of priority between them [69]. Let's start by considering a robot with n degrees of freedom and r tasks, which are defined by

$$\boldsymbol{x}_i = \boldsymbol{f}_i(\boldsymbol{q}) \in \mathbb{R}^{m_i} \qquad \forall i = 1, \ldots, r \tag{5.7}$$

where $\boldsymbol{q}$ is the $n \times 1$ joint displacement vector, $\boldsymbol{x}_i$ is the $m_i \times 1$ task position vector and $\boldsymbol{f}_i()$ a function that defines a generic i-th task constraint of dimension $m_i \leq n$. The differential relation from joint velocities $\dot{\boldsymbol{q}}_i$ to task velocities $\dot{\boldsymbol{x}}_i$ is determined by the Jacobian matrix $\boldsymbol{J}_i(\boldsymbol{q})$ of dimension $m_i \times n$ via

$$\dot{\boldsymbol{x}}_i = \boldsymbol{J}_i(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{5.8}$$

being the task Jacobian equal to

$$\boldsymbol{J}_i(\boldsymbol{q}) = \frac{\partial \boldsymbol{f}_i(\boldsymbol{q})}{\partial \boldsymbol{q}} \tag{5.9}$$

The task constraint $\boldsymbol{f}_i()$ is used to describe r different desired robot functions. For instance, a task could represent the end-effector position and/or orientation, the available joint range, the distance from an obstacle, etc. The hierarchy is defined such that $i = 1$ is top priority, and $i_a < i_b$ implies that i_a has higher priority than i_b. Now, let

$$\boldsymbol{N}_{\mathbf{A}i}(\boldsymbol{q}) = \boldsymbol{I} - \boldsymbol{J}_{\mathbf{A}i}(\boldsymbol{q})^{\dagger}\boldsymbol{J}_{\mathbf{A}i}(\boldsymbol{q}) \tag{5.10}$$

be the projector onto the null space of the augmented Jacobian i.e. the Jacobian matrix that contains all higher-priority Jacobian matrices

$$\boldsymbol{J}_{\mathbf{A}i}(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{J}_1(\boldsymbol{q}) \\ \boldsymbol{J}_2(\boldsymbol{q}) \\ \vdots \\ \boldsymbol{J}_i(\boldsymbol{q}) \end{bmatrix} \tag{5.11}$$

denoting the superscript † the Moore-Penrose pseudo-inverse

$$\boldsymbol{J}_{\mathbf{A}i}(\boldsymbol{q})^{\dagger} = \boldsymbol{J}_{\mathbf{A}i}(\boldsymbol{q})^{T}\left(\boldsymbol{J}_{\mathbf{A}i}(\boldsymbol{q})\boldsymbol{J}_{\mathbf{A}i}(\boldsymbol{q})^{T}\right)^{-1} \tag{5.12}$$

Then, the joint velocity solution can be cast in a recursive fashion

$$\dot{\boldsymbol{q}}_i = \dot{\boldsymbol{q}}_{i-1} + \left(\boldsymbol{J}_i(\boldsymbol{q})\boldsymbol{N}_{\mathbf{A}i}(\boldsymbol{q})\right)^{\dagger}\left(\dot{\boldsymbol{x}}_i - \boldsymbol{J}_i(\boldsymbol{q})\dot{\boldsymbol{q}}_{i-1}\right), \qquad \dot{\boldsymbol{q}}_1 = \boldsymbol{J}_1(\boldsymbol{q})^{\dagger}\dot{\boldsymbol{x}}_1 \tag{5.13}$$

Essentially, this allows the i-th task to be executed with lower priority with respect to the previous $i-1$ tasks since is executed on the null-space projection of the $i-1$ higher priority tasks. That is, the motion is only executed along with those directions in the configuration space not disturbing the higher priority tasks.

The resulting control action provided by the whole-body controller in terms of joint velocities $\dot{\boldsymbol{q}}$ can also be written in closed form as

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_1(\boldsymbol{q})^{\dagger}\dot{\boldsymbol{x}}_1 + \left(\boldsymbol{J}_2\left(\boldsymbol{q}\right)\boldsymbol{N}_{\mathbf{A}1}\left(\boldsymbol{q}\right)\right)^{\dagger}\dot{\boldsymbol{x}}_2 + \cdots + \left(\boldsymbol{J}_r(\boldsymbol{q})\prod_{j=1}^{r-1}\boldsymbol{N}_{\mathbf{A}j}(\boldsymbol{q})\right)^{\dagger}\dot{\boldsymbol{x}}_r \tag{5.14}$$

At this point, we may ask, what if the i-th task is not feasible? If this occurs, we have a task singularity, which means that $\boldsymbol{J}_i(\boldsymbol{q})$ is singular, and the i-th task cannot be satisfied, regardless of all other tasks. However, the great thing about the presented formulation is that remaining lower priority tasks are not affected. This can be easily demonstrated by observing that, if $\boldsymbol{J}_i(\boldsymbol{q})$ is singular, the dimension of the null-space of $\boldsymbol{J}_{\mathbf{A}i}(\boldsymbol{q})$ is not decreased.

5.2.2 Optimization-based WBC

The whole-body control problem can also be expressed as an optimization problem, allowing us to consider not only task constraints in terms of equalities but also inequality constraints. In optimization-based methods, the WBC controller is expressed as cascade optimization problems. Given a set of r constraints with decreasing priority i, these can be described either as linear equalities (5.8) and/or linear inequalities

$$\boldsymbol{A}_i(\boldsymbol{q})\dot{\boldsymbol{q}} \leq \boldsymbol{b}_i \tag{5.15}$$

with $\boldsymbol{A}_i(\boldsymbol{q}) \in \mathbb{R}^{m_i \times n}$ and $\boldsymbol{b}_i \in \mathbb{R}^{m_i}$. The control action $\dot{\boldsymbol{q}}$ that meets all possible task constraints, following the specified hierarchy, can be computed by solving at each level of priority i the following optimization problem [70]

$$\begin{aligned} \mathcal{S}_{i+1} = \quad & \underset{\dot{\boldsymbol{q}} \in \mathcal{S}_i}{\arg\min} \quad \frac{1}{2}\|\boldsymbol{J}_i(\boldsymbol{q})\dot{\boldsymbol{q}} - \dot{\boldsymbol{x}}_i\|^2 + \frac{1}{2}\|\boldsymbol{w}\|^2 \\ & \text{s.t.} \quad \boldsymbol{A}_i(\boldsymbol{q})\dot{\boldsymbol{q}} - \boldsymbol{w} \leq \boldsymbol{b}_i \end{aligned} \tag{5.16}$$

where $\boldsymbol{w}$ is a vector of slack variables, and $\mathcal{S}_i$ corresponds to the set of solutions of $\dot{\boldsymbol{q}}$ satisfying the $i-1$ higher priority task constraints. In the case of kinematic structures with a high number of degrees of freedom, infinite solutions may exist that satisfy $\dot{\boldsymbol{q}} \in \mathcal{S}_i$. Of particular interest then is the solution with minimum norm,

$$\dot{\boldsymbol{q}}^* = \frac{1}{2} \underset{\dot{\boldsymbol{q}} \in \mathcal{S}_i}{\arg\min} \|\dot{\boldsymbol{q}}\|^2 \tag{5.17}$$

A first direct implication of the proposed cascade optimization problems is that

$$\mathcal{S}_{i+1} \subseteq \mathcal{S}_i \tag{5.18}$$

This means that the set of solutions found at a level of priority i is always strictly enforced at lower levels of priority, which is the objective of the WBC hierarchical task strategy.

Additionally, if $\mathcal{S}_i$ is a non-empty convex polytope, $\mathcal{S}_{i+1}$ is also a non-empty convex polytope, which can be described in terms of equality and inequality constraints

$$\forall i,\ \exists \left\{ \widetilde{\boldsymbol{J}}_i(\boldsymbol{q}), \dot{\widetilde{\boldsymbol{x}}}_i, \widetilde{\boldsymbol{A}}_i(\boldsymbol{q}), \widetilde{\boldsymbol{b}}_i \right\} \text{ such that } \dot{\boldsymbol{q}} \in \mathcal{S}_i \iff \begin{cases} \widetilde{\boldsymbol{J}}_i(\boldsymbol{q})\dot{\boldsymbol{q}} = \dot{\widetilde{\boldsymbol{x}}}_i \\ \widetilde{\boldsymbol{A}}_i(\boldsymbol{q})\dot{\boldsymbol{q}} \leq \widetilde{\boldsymbol{b}}_i \end{cases} \tag{5.19}$$

With this representation, (5.16) resorts to a simple Quadratic Program with linear constraints. Overall, in Algorithm 2 we summarize the procedure to perform the prioritized optimization. Figure 5.7 illustrates some examples of how the optimal set is computed for different priority orderings.

Algorithm 2: Whole-body control with task priorities [70]

Input: $\{\boldsymbol{J}_i(\boldsymbol{q}), \dot{\boldsymbol{x}}_i\}_{i=1}^r$ (task equality constraints),
$\{\boldsymbol{A}_i(\boldsymbol{q}), \boldsymbol{b}_i\}_{i=1}^r$ (task inequality constraints)

$\widetilde{\boldsymbol{J}}_1 = \boldsymbol{J}_i(\boldsymbol{q})$; $\widetilde{\boldsymbol{A}}_1(\boldsymbol{q}) = \boldsymbol{A}_1(\boldsymbol{q})$; $\dot{\widetilde{\boldsymbol{x}}}_1 = \dot{\boldsymbol{x}}_1$; $\widetilde{\boldsymbol{b}}_1 = \boldsymbol{b}_i$;
for $i = 1$ **to** r **do**
  $\mathcal{S}_i \equiv \left\{ \widetilde{\boldsymbol{J}}_i(\boldsymbol{q}), \dot{\widetilde{\boldsymbol{x}}}_i, \widetilde{\boldsymbol{A}}_i(\boldsymbol{q}), \widetilde{\boldsymbol{b}}_i \right\}$ = Solution of (5.16);
  $\dot{\boldsymbol{q}}^* = \arg\min_{\dot{\boldsymbol{q}} \in \mathcal{S}_i} \|\dot{\boldsymbol{q}}\|^2$;
  $\widetilde{\boldsymbol{J}}_{i+1}(\boldsymbol{q}) = \begin{bmatrix} \widetilde{\boldsymbol{J}}_i(\boldsymbol{q}) \\ \boldsymbol{J}_i(\boldsymbol{q}) \end{bmatrix}$; $\dot{\widetilde{\boldsymbol{x}}}_{i+1} = \begin{bmatrix} \dot{\widetilde{\boldsymbol{x}}}_i \\ \boldsymbol{J}_i(\boldsymbol{q})\dot{\boldsymbol{q}}^* \end{bmatrix}$;
  $\widetilde{\boldsymbol{A}}_{i+1}(\boldsymbol{q}) = \widetilde{\boldsymbol{A}}_i(\boldsymbol{q})$; $\widetilde{\boldsymbol{b}}_{i+1} = \widetilde{\boldsymbol{b}}_i$;
  foreach $\boldsymbol{a}_{ij}(\boldsymbol{q}) = row_j(\boldsymbol{A_i(q)})$ **do**
    if $\boldsymbol{a}_{ij}(\boldsymbol{q})\dot{\boldsymbol{q}}^* \leq b_{ij}$ **then**
      $\widetilde{\boldsymbol{A}}_{i+1}(\boldsymbol{q}) = \begin{bmatrix} \widetilde{\boldsymbol{A}}_{i+1}(\boldsymbol{q}) \\ \boldsymbol{a}_{ij}(\boldsymbol{q}) \end{bmatrix}$; $\widetilde{\boldsymbol{b}}_{i+1} = \begin{bmatrix} \widetilde{\boldsymbol{b}}_{i+1} \\ b_{ij} \end{bmatrix}$;
    end
    else
      $\widetilde{\boldsymbol{J}}_{i+1}(\boldsymbol{q}) = \begin{bmatrix} \widetilde{\boldsymbol{J}}_i(\boldsymbol{q}) \\ \boldsymbol{a}_{ij}(\boldsymbol{q}) \end{bmatrix}$; $\dot{\widetilde{\boldsymbol{x}}}_{i+1} = \begin{bmatrix} \dot{\widetilde{\boldsymbol{x}}}_i \\ \boldsymbol{a}_{ij}(\boldsymbol{q})\dot{\boldsymbol{q}}^* \end{bmatrix}$;
    end
  end
end
return: $\dot{\boldsymbol{q}}^*$ (joint velocities);

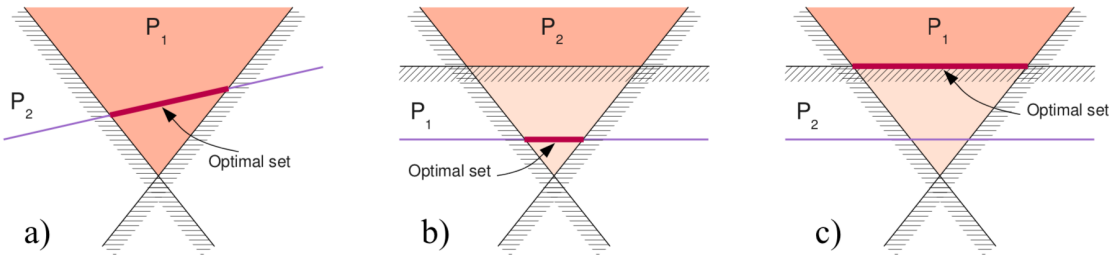

Figure 5.7: Illustration of the optimal sets for prioritization problems involving both linear equality and inequality constraints. a) The constraint of two linear inequalities P_1 and equality constraint P_2 have common solutions so the priority does not matter. b) Equality P_1 has priority over the inequality constraints P_2. The error is minimized with respect to inequalities that could not be satisfied. c) Inequalities P_1 have priority over equality P_2. The optimal set minimizes the distance to the equality set (after [70]).

5.3 Variable Admittance Control

The type of robots to which human motion is usually transferred is designed to coexist and cooperate with people in different applications within the same workspace. In these environments physical human-robot interaction (pHRI) might occur, becoming safety a key issue. The robot has to feature a certain degree of active compliance. A popular method for achieving this is admittance control. By measuring the interaction forces, the set-point to a low-level motion controller is changed through a virtual spring-mass-damper model dynamics to achieve some preferred interaction responsive behavior.

Reproduction of human motion involves two competing control objectives. On the one hand, an accurate position control (high stiffness) is desirable. On the other hand, when physical human-robot interaction occurs, compliance (low stiffness) is of vital importance to ensure safety. This problem can be addressed with a variable admittance control scheme that adaptively modulates the robot dynamics; switching between stiff and compliant behaviors based on the external forces, changing the virtual spring-mass-damper dynamics continuously during the task. However, this has important implications on the stability properties of the control system which cannot be overlooked.

We start, in Section 5.3.1, providing an intuitive introduction to admittance control in order to grasp the main ideas. Afterward, in Section 5.3.2, we discuss some stability considerations when adopting a variable admittance control scheme. Finally, in Section 5.3.3, we present a role-adaptive admittance controller for dynamically switching the robot's behaviour, ensuring both a tight precision control while reproducing the human motion, and compliance during physical human-robot interaction.

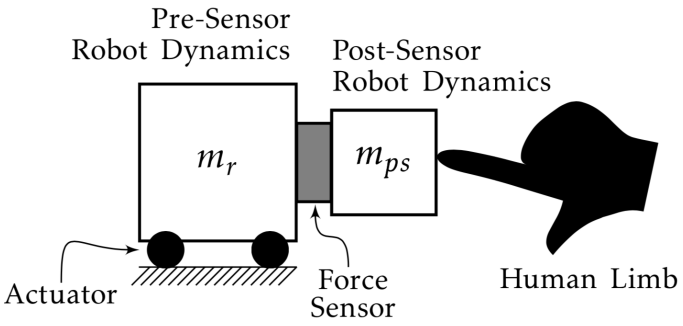

Figure 5.8: Generic overview of the mechanical system. An actuator moves all mechanics (robot inertia m_r and some dissipation) placed before the force sensor. Behind the force sensor, there will be mechanics that generate force sensor measurements during motion (m_{ps}). These post-sensor mechanics also interact with the human limb. The sensor is assumed to be infinitely stiff and its mass is absorbed in m_{ps} (after [71]).

5.3.1 Simple Admittance Control Model

The goal of this section is to give the reader an introduction to admittance controller design. We study a generic mechanical set-up [71] and a control model to explain the main concepts behind admittance control in the context of physical human-robot interaction. A scheme of the system is shown in Figure 5.8. The actuator imposes forces on the mechanics of the device, which can consist of robotic links. Close to the interaction point a sensor measures the forces exerted by the human. The admittance controller should attempt to respond to these forces according to some specified virtual dynamics.

From the control perspective, the robot is equivalent to a rigid body mass with some dissipation. This robot can be in contact with a human that applies a force F_{ext}, which can be from human impedance (shown in dotted gray in Figure 5.9). The equation of motion of the system, omitting the human impedance, absorbing any external force into $F_{ext}(t)$ is given by

$$(m_r + m_{ps})\ddot{x}(t) + b_r\dot{x}(t) = F_{ext}(t) + F_c(t) \tag{5.20}$$

with m_r the pre-sensor robot inertia and m_{ps} the post-sensor robot inertia, $x(t)$ the real robot position, b_r the viscous effects in the drive train, and $F_c(t)$ the force applied by the controller through the actuators. Now, we impose the following dynamic admittance model

$$F_{ext}(t) = m_d\ddot{x}(t) + b_d\dot{x}(t) + k_d\left(x(t) - x_d\right) \tag{5.21}$$

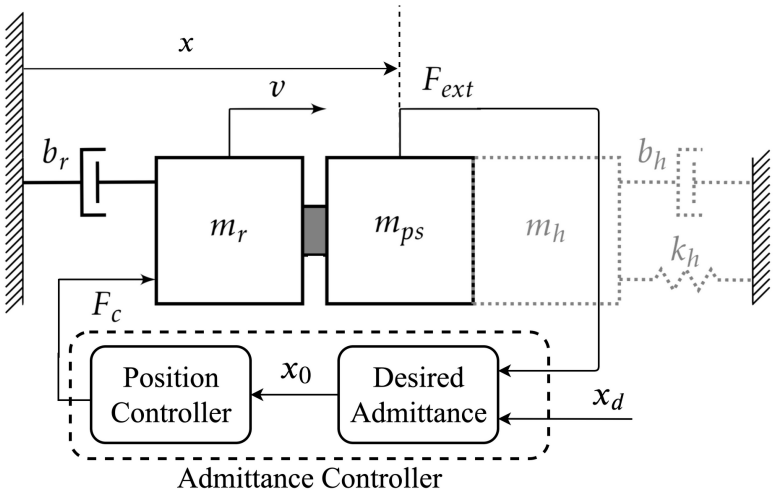

Figure 5.9: Rigid robot from a control perspective. An external force F_{ext} and a controller force from an actuator F_c are applied to the robot inertia m_r combined with the post-sensor inertia m_{ps}, both resulting in some robot velocity v. Some energy losses during robot motion are modeled as viscous damping b_r. The robot can be rigidly connected to a human with inertia m_h, stiffness k_h, and damping b_h, shown by the gray dotted outline. By measuring F_{ext} the set-point to a low-level position controller x_0 is adjusted according to the desired admittance (after [71]).

where $m_d > 0$, $b_d > 0$, $k_d > 0$ and x_d are the desired virtual inertia, damping, stiffness and robot position respectively. Note that the presence of an external force $F_{ext} \neq 0$ prevents the equilibrium to be at the desired robot position $x_0 = x_d$. Indeed, assuming F_{ext} constant and substituting $\ddot{x}(t) = \dot{x}(t) = 0$ in (5.21), the equilibrium position is given by

$$x_0 = \frac{F_{ext}}{k_d} + x_d \tag{5.22}$$

As we can see the higher the value of $k_d \to \infty$ (i.e. stiffer), the closer the equilibrium point is to the desired position $x_0 \to x_d$. Inversely, the lower the value of k_d the more compliant the robot is to the external force F_{ext}, allowing a greater position error. The remaining parameters m_d and b_d affect the temporal evolution towards the equilibrium position as for a damped harmonic oscillator. As a rule of thumb for robotic applications, the following inequality should be fulfilled

$$b_d \geq 2\sqrt{m_d k_d} \tag{5.23}$$

otherwise, we can have oscillations in the transient. Our strategy for the controller design is now to implement relation (5.21) via admittance control, using a position controller to achieve the desired end-effector dynamics (Figure 5.9).

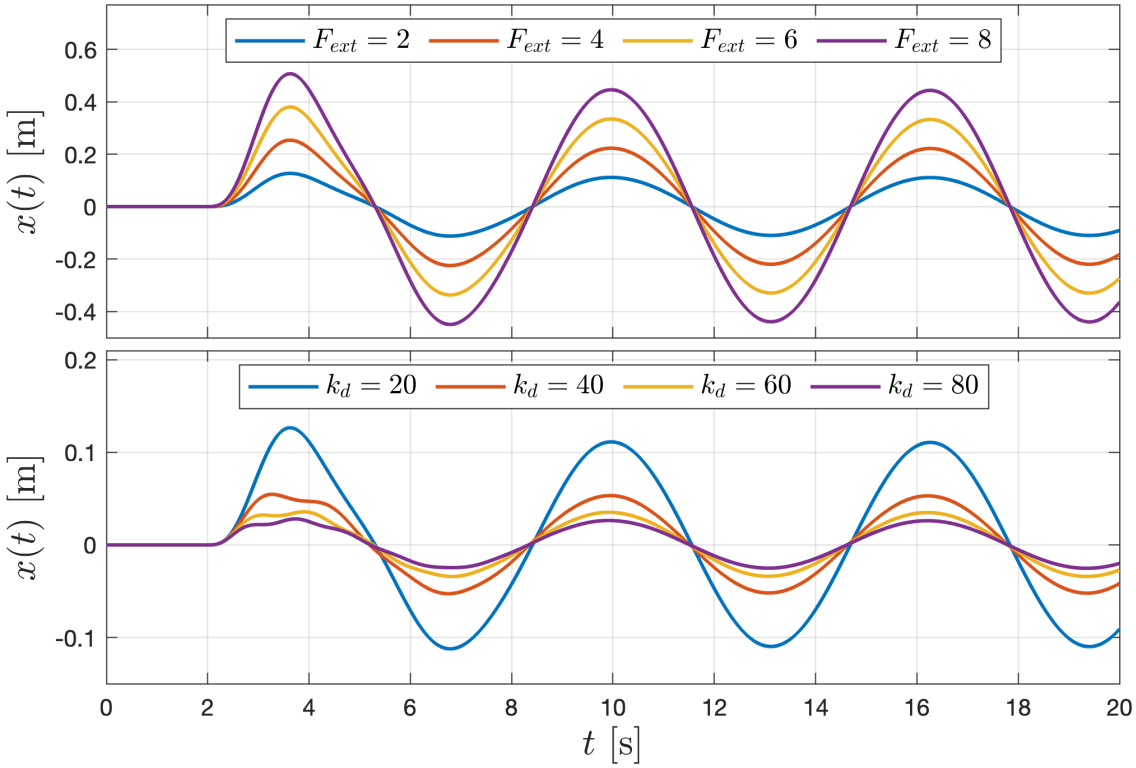

Figure 5.10: Closed-loop simulation of the mechanical system in Figure 5.9 with the parameters in Table 5.1. The desired position is $x_d = 0$, and from $t = 2$ s a sinusoidal external force F_{ext} is exerted on the system. On top, the evolution of the position $x(t)$ for different amplitudes of the external force F_{ext}. Below, applying an external force with amplitude $F_{ext} = 2$ N, the evolution of $x(t)$ for different values of the stiffness k_d.

In order to analyze the behavior of the resulting closed-loop system, we have to consider a position control law. We can, for instance, use a simple proportional-derivative (PD) controller of the form [72]

$$F_c(t) = -P\left(x(t) - x_0(t)\right) - D\left(\dot{x}(t) - \dot{x}_0(t)\right) \tag{5.24}$$

with positive proportional and derivative gains $P > 0$ and $D > 0$. Assuming the parameters for the mechanical system in Table 5.1, we can see some simulations of the closed-loop system in Figure 5.10. Setting a constant desired position $x_d = 0$, we can observe the effect of the external force $F_{ext}(t)$ magnitude and the admittance stiffness k_d on the evolution of the robot position $x(t)$. Using the proposed admittance control scheme we can see that the robot presents a compliant behavior. The dynamics of this behavior can be adjusted by means of the desired admittance (5.21). We can observe that for higher values of k_d we achieve a tighter position control, while for smaller values of k_d we are more compliant to the external force, allowing a greater deviation from the desired position $x_d = 0$ as $F_{ext}(t)$ increases.

Parameter	Value	Units
m_d	2	kg
b_d	2	Ns/m
k_d	20	N/m
m_r	10	kg
m_{ps}	2	kg
b_r	5	Ns/m
P	100	N/m
D	2000	Ns/m

Table 5.1: Parameters for the closed-loop simulation of the system in Figure 5.9.

5.3.2 Stability in Variable Admittance Control

The interaction forces are subject to uncertainties when robots are operating in human-shared environments. The desired response can be adaptively regulated using a variable admittance control scheme. However, careful attention should be paid to the design of the admittance profile, since it has important implications on the stability properties of the system. In this section, we focus on studying the stability constraints following the procedure proposed in [73].

In variable admittance control, the objective is to maintain the following dynamic relationship between the external force $\boldsymbol{F}_{ext}(t)$ and the robot's end-effector position error $\boldsymbol{e}(t)$

$$\boldsymbol{M}(t)\ddot{\boldsymbol{e}}(t) + \boldsymbol{B}(t)\dot{\boldsymbol{e}}(t) + \boldsymbol{K}(t)\boldsymbol{e}(t) = \boldsymbol{F}_{ext}(t) \tag{5.25}$$

where $\boldsymbol{M}(t)$, $\boldsymbol{B}(t)$ and $\boldsymbol{K}(t)$ denote the desired virtual inertia, damping, and stiffness profiles respectively. These determine the behavior of the robot when subjected to an interaction force $\boldsymbol{F}_{ext}(t)$. If $\boldsymbol{M}$, $\boldsymbol{B}$ and $\boldsymbol{K}$ are constant, the system is asymptotically stable for any symmetric positive definite choice of the matrices. However, we are concerned with varying admittance control. Without loss of generality, we assume that $\boldsymbol{M}$ remains constant while $\boldsymbol{B}(t)$ and $\boldsymbol{K}(t)$ are time-varying functions.

The stability properties of (5.25), are commonly performed based on energy considerations since the dynamics allow a physical interpretation analogous to a classic mass-spring-damper system. Consider the following Lyapunov candidate function

$$V_1(t) = \frac{1}{2}\dot{\boldsymbol{e}}(t)^t \boldsymbol{M}\dot{\boldsymbol{e}}(t) + \frac{1}{2}\boldsymbol{e}(t)^T \boldsymbol{K}(t)\boldsymbol{e}(t) \tag{5.26}$$

Differentiating $V_1(t)$ along the trajectories of (5.25) with $\boldsymbol{F}_{ext} = \mathbf{0}$ and $\boldsymbol{M}$ constant, we obtain

$$\dot{V}_1(t) = -\dot{\boldsymbol{e}}(t)^T \boldsymbol{B}(t)\dot{\boldsymbol{e}}(t) + \boldsymbol{e}(t)^T \dot{\boldsymbol{K}}(t)\boldsymbol{e}(t) \tag{5.27}$$

Equation (5.27) is negative semidefinite for a negative semidefinite $\dot{\boldsymbol{K}}(t)$. Hence, we can conclude stability at the origin only if all the eigenvalues of the stiffness matrix are either constant or decreasing. Assuming $\boldsymbol{e}(t) \neq 0$, increasing the stiffness eigenvalues can inject potential energy into the system, and it is hence intuitively clear that this practice can cause unstable behavior.

By inspection of the expression in (5.27), the obvious solution to the problem would be to design a controller that tries to follow the desired stiffness profile as well as possible, but limiting it when (5.27) becomes positive. However, such an approach has several disadvantages, the most important being that the admissible stiffness profile depends on the state of the robot and can hence not be known beforehand.

Experience of varying stiffness control suggests that in general, reasonable varying stiffness profiles show no destabilization tendencies. This motivates the search for a less conservative Lyapunov candidate function than (5.26). Consider the following Lyapunov candidate function

$$V_2(t) = \frac{\left(\dot{\boldsymbol{e}}(t) + \alpha \boldsymbol{e}(t)\right)^T \boldsymbol{M} \left(\dot{\boldsymbol{e}}(t) + \alpha \boldsymbol{e}(t)\right)}{2} + \frac{\boldsymbol{e}(t)^T \boldsymbol{\beta}(t) \boldsymbol{e}(t)}{2} \tag{5.28}$$

where

$$\boldsymbol{\beta}(t) = \boldsymbol{K}(t) + \alpha \boldsymbol{B}(t) - \alpha^2 \boldsymbol{M} \tag{5.29}$$

with some positive constant α chosen, such that $\boldsymbol{\beta}(t)$ is positive semidefinite for all $t > 0$. Differentiating $V_2(t)$ along the trajectories of (5.25) with $\boldsymbol{F}_{ext}(t) = \mathbf{0}$ yields

$$\dot{V}_2(t) = \dot{\boldsymbol{e}}(t)^T \left(\alpha \boldsymbol{M} - \boldsymbol{B}(t)\right) \dot{\boldsymbol{e}}(t) + \boldsymbol{e}(t)^T \left(\frac{1}{2}\dot{\boldsymbol{K}}(t) + \frac{\alpha}{2}\dot{\boldsymbol{B}}(t) - \alpha \boldsymbol{K}(t)\right) \boldsymbol{e}(t) \tag{5.30}$$

Therefore, assuming $\boldsymbol{M}$ constant, $\boldsymbol{K}(t)$ and $\boldsymbol{B}(t)$ continuously differentiable, and the three of them symmetric, positive definite matrices, the system (5.25) is globally uniformly stable if there exists $\alpha > 0$ such that $\forall t \geq 0$:

1. $\alpha \boldsymbol{M} - \boldsymbol{B}(t)$ is negative semidefinite.
2. $\dot{\boldsymbol{K}}(t) + \alpha \dot{\boldsymbol{B}}(t) - 2\alpha \boldsymbol{K}(t)$ is negative semidefinite.

In practice, the admittance parameter that has the most significant impact on task performance is the stiffness $\boldsymbol{K}(t)$. Hence, it is often reasonable to give priority to the stiffness design. The damping can then be chosen to guarantee that the desired stiffness profile can be stably executed, typically by using critical damping. Inspection of the stability constraints reveals that the least conservative constraints are given by α chosen so the largest eigenvalue of $\alpha\boldsymbol{M} - \boldsymbol{B}(t)$ remains negative during the task. Hence, to have the least conservative constraints,

$$\alpha = \min_t \frac{\underline{\lambda}\left(\boldsymbol{B}(t)\right)}{\overline{\lambda}\left(\boldsymbol{M}\right)} \tag{5.31}$$

where $\underline{\lambda}()$ and $\overline{\lambda}()$ denote the largest and smallest eigenvalue, respectively. For robotic applications, these conditions can typically be simplified to a form that has a more intuitive interpretation. Usually, $\boldsymbol{M}$, $\boldsymbol{D}(t)$ and $\boldsymbol{K}(t)$ are diagonal matrices. Therefore, system (5.25) can be uncoupled in independent scalar systems:

$$m\ddot{e}(t) + b(t)\dot{e}(t) + k(t)e(t) = F_{ext}(t) \tag{5.32}$$

Assuming $b(t) = 2\zeta\sqrt{mk(t)}$, where $\zeta > 0$ is a constant damping ratio, and substituting $\dot{b}(t)$ into the second stability condition, yields the following upper bound for the stiffness derivative

$$\dot{k}(t) < \frac{2\alpha\sqrt{k(t)}^3}{\sqrt{k(t)} + 2\zeta\alpha\sqrt{m}} \tag{5.33}$$

which taking the least conservative constraints for α (5.31) results

$$\dot{k}(t) < \frac{4\zeta\sqrt{k(t)}^3\sqrt{\max_t k(t)}}{\sqrt{mk(t)} + 4\zeta^2\sqrt{\max_t k(t)}} \tag{5.34}$$

Note that stability is favored by smooth variations and large values of the stiffness profile $k(t)$, and vice-versa.

5.3.3 Role Adaptive Admittance Controller

The classical approach for dealing with physical human-robot interaction typically involves the robot yielding compliantly to the motion of the human. However, when a robot is reproducing a demonstrated motion, a compliant behavior can degrade the performance of the position control due to disturbances on the external force sensors. On the other hand, following tightly the demonstrated trajectories can be dangerous when operating in human-shared environments.

What we are looking for is an adaptive varying admittance profile that adjusts the role of the robot dynamically. Understanding by the role the balance between the tracking precision (leader), and compliance (follower). In order to switch the robot role between a follower, with low stiffness k_{min} and leader, with high stiffness k_{max}, we propose the following varying stiffness profile

$$k(t) = k_{min} + \gamma(t)\,(k_{max} - k_{min}) \tag{5.35}$$

where $\gamma(t) \in [0, 1]$ is a scalar role adaptation factor. The value of $\gamma(t)$ should be derived from the interaction force feedback $F_{ext}(t)$ in order to achieve the desired responsive behavior. Continuous and smooth variations of the stiffness profile, show no destabilization tendencies. Therefore, we propose the following role factor sigmoid-like profile, which is infinitely differentiable

$$\gamma(t) = 1 - \frac{1}{1 + \exp\left(-c_1\left(\psi(t) - c_2\right)\right)} \tag{5.36}$$

where $c_1 > 0$ and $c_2 \in [0, 1]$ are design parameters. The first one determines the speed of the transition between stiff and compliant behaviors, while the second one specifies at which value of $\psi(t)$ does this transition occur. On the other hand, $\psi(t) \in [0, 1]$ is an interaction factor that varies according to the external force feedback $F_{ext}(t)$. We propose the following interaction factor dynamics

$$\dot{\psi}(t) = \begin{cases} c_3 & \text{if } F_{ext}(t) > F_{thresh} \text{ and } \psi(t) \leq 1 \\ -c_4 & \text{if } F_{ext}(t) \leq F_{thresh} \text{ and } \psi(t) \geq 0 \\ 0 & \text{else} \end{cases} \tag{5.37}$$

where F_{thres} is the force threshold to consider that physical human-robot interaction is occurring, and $c_3 > 0$ and $c_4 > 0$ are design parameters. Essentially, the idea of the proposed variable admittance profile is that, if a force $F_{ext}(t) > F_{thresh}$ is detected on the end effector, the interaction factor increases $\psi(t) \to 1$, making the role adaptation factor decrease $\gamma(t) \to 0$, evolving the stiffness smoothly towards $k(t) \to k_{min}$. Inversely, if $F_{ext}(t) < F_{thresh}$ we have that $k(t) \to k_{max}$.

Now, we can derive the stability constraints for our role-adaptive variable admittance profile, using the results of Section 5.3.2. Substituting (5.36) into (5.35) and differentiating we obtain the following upper bound for the stiffness derivative

$$\dot{k}(t) = \frac{-c_1 \exp\left(-c_1\left(\psi(t) - c_2\right)\right)\dot{\psi}(t)}{\left(1 + \exp\left(-c_1\left(\psi(t) - c_2\right)\right)\right)^2}\left(k_{max} - k_{min}\right) \leq \frac{c_1 c_4}{4}\left(k_{max} - k_{min}\right) \tag{5.38}$$

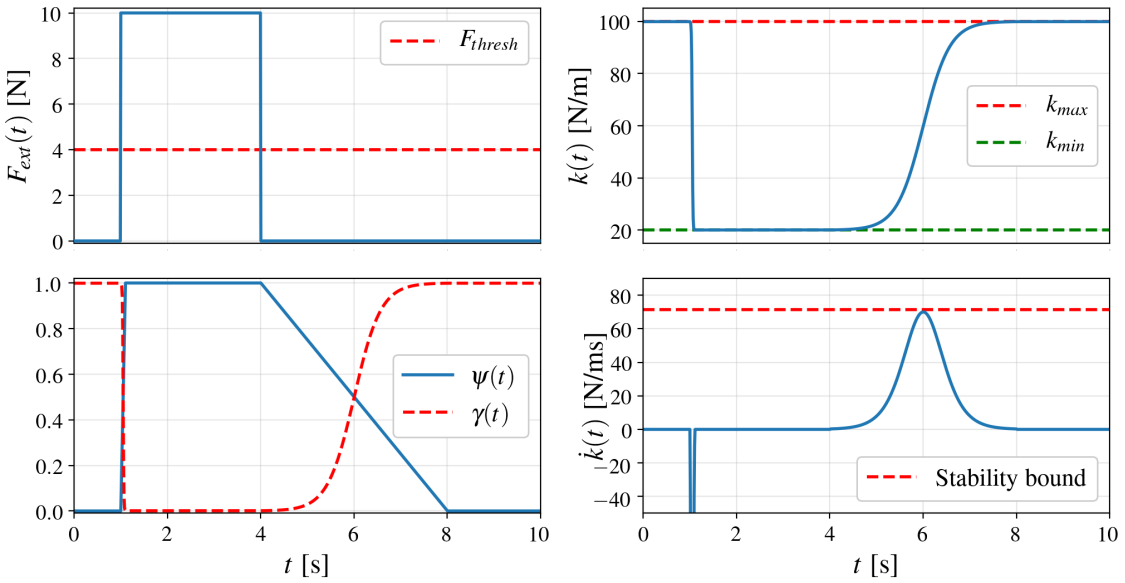

Figure 5.11: Behaviour of the proposed role adaptive admittance controller with parameters $c_1 = 14$, $c_2 = 0.5$, $c_3 = 10$ and $c_4 = 0.25$ when the force depicted in the upper-left plot is applied. Below, we can see the evolution of the role adaptation factor $\gamma(t)$ (5.36) and the interaction factor $\psi(t)$ (5.37). On the upper-right plot we can observe how does the stiffness $k(t)$ vary (5.35), switching between k_{min} and k_{max} according to $F_{ext}(t)$. Below we show its derivative (5.38) along with the sufficient stability bound (5.40).

Then, for the proposed variable admittance profile we can also compute a lower bound for the right-hand side of (5.34)

$$\frac{4\zeta\sqrt{k(t)}^3\sqrt{\max_t k(t)}}{\sqrt{mk(t)} + 4\zeta^2\sqrt{\max_t k(t)}} \geq \frac{4\zeta\sqrt{k_{min}}^3}{\sqrt{m} + 4\zeta^2} \tag{5.39}$$

Finally, substituting the bound of the stiffness derivative (5.38) and (5.39) in the stability condition (5.34) we can derive the following sufficient stability condition

$$\frac{c_1 c_4}{4}\left(k_{max} - k_{min}\right) < \frac{4\zeta\sqrt{k_{min}}^3}{\sqrt{m} + 4\zeta^2} \tag{5.40}$$

The great thing about this sufficient condition is that is independent of the state and can be computed beforehand. By tuning the design parameters adequately we can ensure stability. From inspection of (5.40) and the example shown in Figure 5.11 we can study several interesting properties of the proposed variable admittance controller regarding its stability. The most relevant perhaps is that instabilities can only occur in the transition from compliant (k_{min}) to stiff (k_{max}) dynamics. This means that, when a person physically interacts with the robot, it can almost react instantaneously without compromising the system stability.

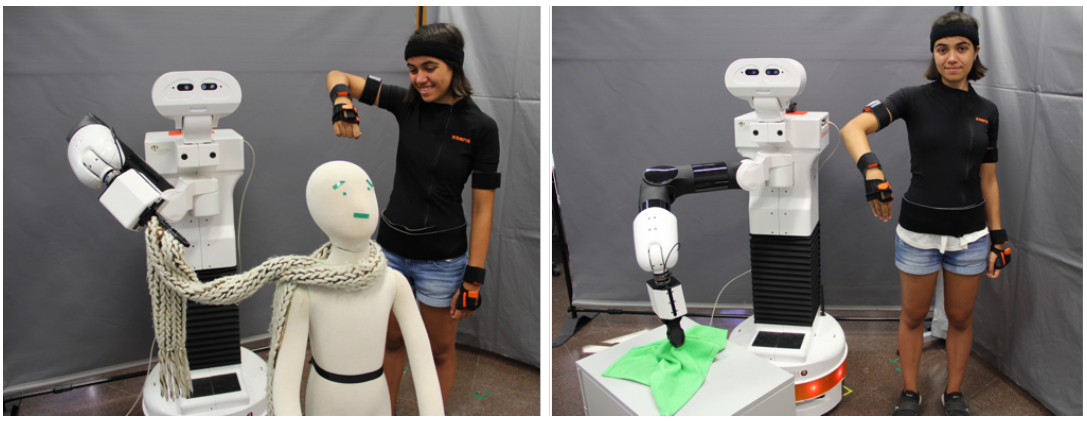

Figure 5.12: Transferring human motion to robots while ensuring safe human-robot physical interaction can be a powerful and intuitive tool for teaching assistive tasks such as helping people with reduced mobility to get dressed or cleaning.

Rapid transitions from the leader to the follower role can be achieved for high values of the design parameter c_3, which we can see that does not appear in (5.40). For favouring stability small values of $(k_{max} - k_{min})$ are preferred. Note that for the limit case $k_{max} = k_{min}$ stability in ensured regardless of the value of all other parameters. Stability is also supported for small values of c_1 and c_4, which basically implies a slower transition from k_{min} to k_{max}.

As a final concluding remark, keep in mind that stability condition (5.40) is sufficient but not necessary. That is, if the design parameters fulfill the condition we can ensure stability. However, if this is not the case, we cannot conclude that the system is unstable.

5.4 Whole-Body Motion Transfer with TIAGo

Transferring human motion to a mobile robotic manipulator and ensuring safe physical human-robot interaction are crucial steps towards automating complex manipulation tasks in human-shared environments. As we have discussed throughout this chapter, this raises several challenges in robotics.

In this section, we present a novel human to robot whole-body motion transfer framework for the TIAGo robot [74]. The main contributions are the formulation of a solution to the correspondence problem, and the definition of a control scheme for effective real-time whole-body imitation, while ensuring compliance and stability during physical human-robot interaction. The presented framework is validated through several real-world experiments with the TIAGo robot. Results show effective real-time imitation and dynamic behavior adaptation, opening the door for intuitive human motion transfer to service robots (Figure 5.12).

Figure 5.13: Overview of the main components of the TIAGo robot.

We start, in Section 5.4.1, by giving a brief overview of the TIAGo robot. Next, in Section 5.4.2, we describe the Xsens motion capture system, our interface for providing demonstrations in real-time. Afterward, in Section 5.4.3, we look into the solution for the correspondence problem. Then, in Section 5.4.4, we present a whole-body control system for achieving real-time imitation with TIAGo's upper body and base, while ensuring compliance. Finally, in Section 5.4.5, we evaluate the performance of the proposed framework through several real-world experiments.

5.4.1 TIAGo Robot

TIAGo is a mobile service robot designed to work in indoor environments. It is developed by PAL Robotics, a company with its corporate headquarters located in Barcelona. The TIAGo robot is equipped with an extendable torso and a manipulator's arm to grab tools and objects. Its sensor suite allows it to perform a wide range of perception, manipulation, and navigation tasks. An overview of its main components is depicted in Figure 5.13.

The upper body of TIAGo is composed of a lifting torso with a stroke of 35 cm. The robot height can be adjusted between 110 and 145 cm. The frontal part is equipped with a stereo microphone and a speaker, both aimed at human-robot interaction. The arm of TIAGo has 7 degrees of freedom (DoF), with a force/torque sensor attached at the endpoint of the wrist. The head has 2 DoF, providing pan-tilt movements, and it is equipped with an RGB-D camera. The mobile base has a differential drive mechanism and is provided with a laser plane for SLAM and safety purposes. On the rear part of the mobile base, there are three ultrasound sensors to prevent collisions when moving backward [75].

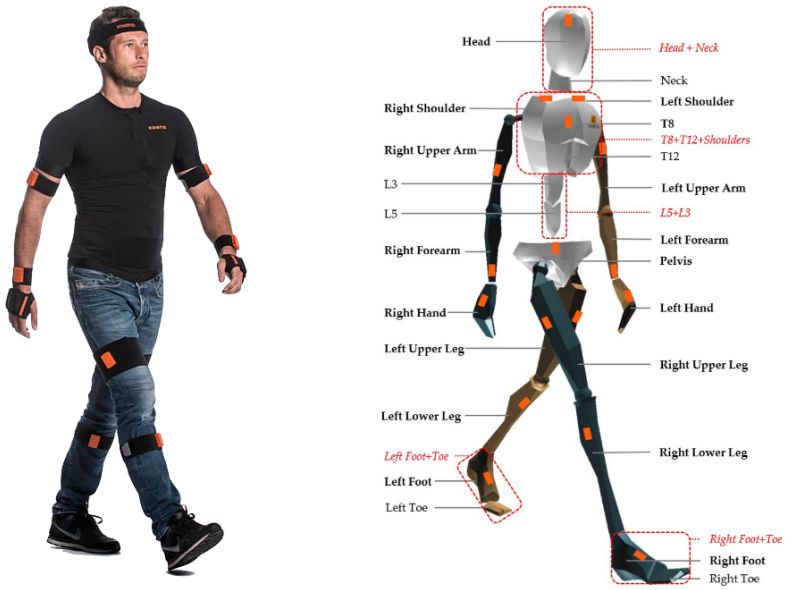

Figure 5.14: Xsens MVN consists of 17 inertial and magnetic motion trackers. Data from wireless trackers is transmitted to a PC where it is further processed and visualized.

5.4.2 Xsens MVN Motion Capture

The Xsens MVN inertial motion capture system is an easy-to-use interface for full-body human motion capture [76]. We use it for transferring the demonstrator's movements online to the TIAGo robot. It is based on small, unobtrusive inertial and magnetic sensors combined with advanced algorithms and biomechanical models. Compared to alternative motion capture systems based on external emitters and/or cameras, inertial motion capture does not rely on any external infrastructure allowing it to be used anywhere.

Xsens MVN captures the motion of the human body using 17 motion trackers that are attached to the feet, lower legs, upper legs, pelvis, shoulders, sternum, head, upper arms, forearms, and hands, as shown in Figure 5.14. A custom lycra suit with dedicated zippers and body straps is provided for a simplified mounting of motion trackers at specific body locations. The sensor modules are inertial and magnetic measurement units that contain gyroscopes, accelerometers, and magnetometers. Each module runs an advanced signal processing pipeline that ensures 3D tracking accuracy.

The key element of Xsens MVN is the software engine, where the data of the individual motion trackers is combined with biomechanical models of the human body to obtain segment positions and orientations. It is capable of sending real-time motion capture data of 23 body segments using the UDP/IP communication protocol.

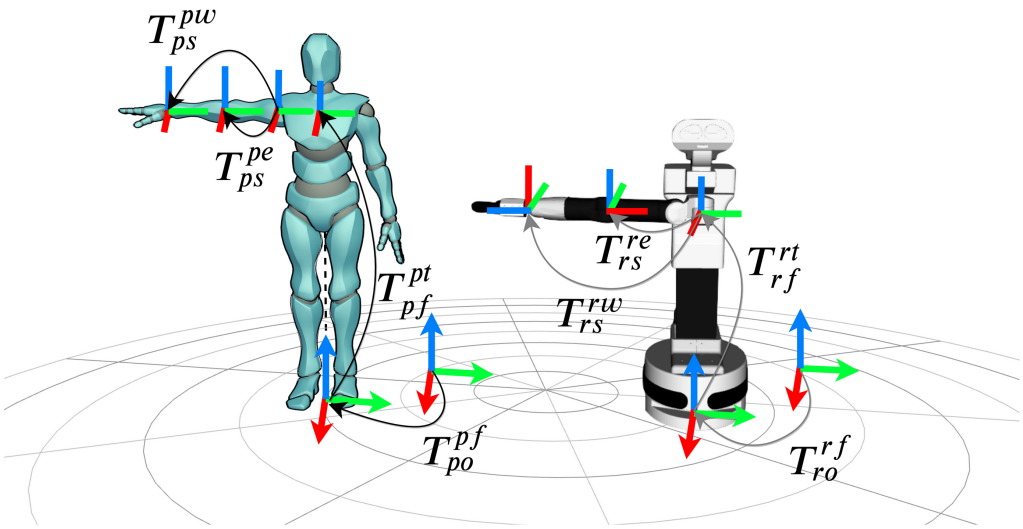

Figure 5.15: Mapping in Cartesian space of an equivalent pose between a human model and the TIAGo robot. The colors red, green, and blue are the x, y, and z axes respectively of each reference frame attached to a body segment. Note that for the particular case of the TIAGo robot the torso and the shoulder frames are equivalent.

5.4.3 Correspondence Problem Solution

Using the Xsens MVN motion capture system as the interface to provide the demonstrations to the TIAGo robot, the correspondence problem can be addressed as discussed in Section 5.1.3. Adopting the same notation, we have $\mathcal{O} \equiv SE(3)^{23}$, that is, the human movements are encoded as the trajectory of a 23-dimensional array (one element per each tracked body segment) of three-dimensional Euclidean groups. Each element is represented by a homogeneous transformation from reference a to b, which can be written as

$$\boldsymbol{T}_a^b = \begin{bmatrix} \boldsymbol{R}_a^b & \boldsymbol{p}_a^b \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{5.41}$$

where $\boldsymbol{R}_a^b \in SO(3)$ and $\boldsymbol{p}_a^b \in \mathbb{R}^3$ are the rotational and translational components of the transform, respectively.

The objective now is to define a complete mapping in the form of (5.6). In particular, we need to specify $f_{\mathcal{O}}^{\mathcal{G}}$ (5.2), that is, the equivalence between the observed human motion and the robot's desired behavior. We consider that the human pose and the robot pose are equivalent if the relative position and orientation of the person's right (or left) wrist, elbow, chest, and the projection of the pelvis onto the floor with respect to an arbitrarily fixed reference frame are as close as possible to those equivalent links of the robot up to a scaling factor in Cartesian space, as depicted in Figure 5.15.

Given the transforms $\boldsymbol{T}_{po}^{pf}$, $\boldsymbol{T}_{pf}^{pt}$, $\boldsymbol{T}_{ps}^{pe}$ and $\boldsymbol{T}_{ps}^{pw}$, where po, pf, pt, ps, pe and pw stand for the person arbitrary origin, virtual footprint (i.e. projection of the pelvis onto the floor), torso, shoulder, elbow and wrist reference frames respectively, and a sample equivalent person-robot pose, like the one depicted in Figure 5.15. The correspondent robot pose is fully-determined by $\boldsymbol{T}_{ro}^{rf}$, $\boldsymbol{T}_{rf}^{rt}$, $\boldsymbol{T}_{rs}^{re}$ and $\boldsymbol{T}_{rs}^{rw}$, where ro, rf, rt, rs, re, rw are the equivalent robot links (Figure 5.15). The rotational components of these robot transforms are defined as

$$\boldsymbol{R}_{ra}^{rb} = \boldsymbol{Rs}_{pa}^{ra} \cdot \boldsymbol{R}_{pa}^{pb} \tag{5.42}$$

where $\boldsymbol{Rs}_{pa}^{ra}$ is the rotational component of the relative transform between the person's frame pa and the robot's equivalent frame ra when both are in the equivalent sample pose. On the other hand, their translational components are given by

$$\boldsymbol{p}_{ro}^{rf} = \boldsymbol{p}_{po}^{pf} \tag{5.43}$$

$$\boldsymbol{p}_{rf}^{rt} = L_{rf}^{rt} \cdot \frac{\boldsymbol{p}_{pf}^{pt}}{\|\boldsymbol{p}_{pf}^{pt}\|} \tag{5.44}$$

$$\boldsymbol{p}_{rs}^{re} = L_{rs}^{re} \cdot \frac{\boldsymbol{p}_{ps}^{pe}}{\|\boldsymbol{p}_{ps}^{pe}\|} \tag{5.45}$$

$$\boldsymbol{p}_{rs}^{rw} = \boldsymbol{p}_{rs}^{re} + L_{re}^{rw} \cdot \frac{\boldsymbol{p}_{ps}^{pw} - \boldsymbol{p}_{ps}^{pe}}{\|\boldsymbol{p}_{ps}^{pw} - \boldsymbol{p}_{ps}^{pe}\|} \tag{5.46}$$

where L_{rf}^{rt} is the robot's base to torso height when the torso is fully extended, L_{rs}^{re} and L_{re}^{rw} are the lengths of the robot's equivalent shoulder to elbow and elbow to wrist segments respectively. Note that essentially, the translations are obtained using a generalized scaling model. At this point, since we have fully determined the robot pose from the observations of the demonstrator's motion, a complete definition of $f_{\mathcal{O}}^{\mathcal{G}}$, and consequently a solution to the correspondence problem, is provided.

5.4.4 Whole-Body Control System

The reproduction of human motion with a complex mobile manipulator robot-like TIAGo requires tight control of the multiple degrees of freedom. We propose a control architecture that decouples the upper body and the mobile base. An overview is depicted in Figure 5.16. For ensuring both, accurate position control and safety while reproducing the upper body motion, we combine the role adaptive variable admittance controller derived in Section 5.3.3 with an optimization-based WBC analogous to the one presented in Section 5.2.2. On the other hand, for imitating the human walking motion with the mobile base, we implement a control algorithm at a velocity level that efficiently handles the nonholonomic constraints imposed by the differential drive mechanism.

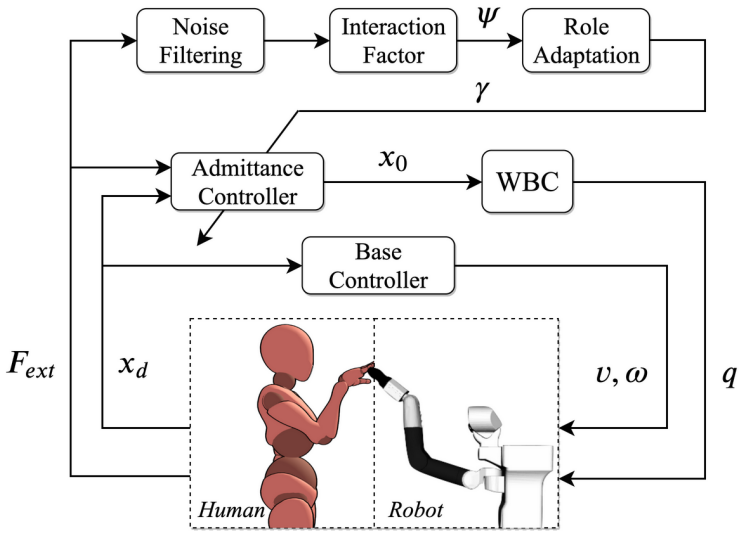

Figure 5.16: Overview of the whole-body human to robot motion transfer control scheme for the TIAGo robot. The human operator provides a reference pose $\boldsymbol{x}_d$ for the robot to imitate and possibly exerts a force $\boldsymbol{F}_{ext}$ on the robot's end-effector. Based on $\boldsymbol{x}_d$ and $\boldsymbol{F}_{ext}$, the role adaptive admittance controller generates a new setpoint $\boldsymbol{x}_0$ for the robot's upper body links. The robot is then driven towards the required configuration $\boldsymbol{q}$ using a hierarchical WBC controller. The mobile base is controlled separately at a velocity level (v, ω) for efficient handling of the nonholonomic constraints while imitating the human walking motion, encoded also in $\boldsymbol{x}_d$.

Upper Body Control

Based on the solution of the correspondence problem for the TIAGo robot derived in Section 5.4.3, the robot needs to reach multiple varying goals in Cartesian space simultaneously for imitating the human's upper body motion (Figure 5.15). This makes WBC a suitable control framework since the multiple goals can be defined as a set of tasks with an adequate hierarchy such as they do not interfere with each other. We use PAL robotics' implementation, which is based on the Stack of Tasks [77]. Taking into consideration the equivalence human-robot relations discussed in the previous section, we select the following task hierarchy:

1. Joint limit avoidance
2. Self-collision avoidance
3. End-effector pose admittance control
4. Torso position control
5. Elbow pose control

The first task addresses the physical limitations and restrictions of the robotic system. Thus, it should always be active. The second and third tasks are concerned with safety aspects and for this reason, they must be at a high priority to prevent dangerous situations. Defining the end-effector task with a high priority we also ensure a correct end-effector goal tracking, which is of critical importance for manipulation tasks. The torso task guarantees that the base of the robot's arm is at the required height. Finally, with the elbow task, we achieve not just a correct positioning of the end-effector but the imitation of the whole arm's posture. For a robot with human-like affordances, which is the case of the TIAGo robot, a correct imitation of the human motion can be achieved with this hierarchy.

Differential Drive Base Control

TIAGo's mobile base is controlled by a differential drive mechanism, which consists of two separately driven wheels placed on either side of the robot base. Direction is changed by varying the relative rate of rotation of its wheels. These types of mobile bases are usually controlled at a velocity level. Let $(x, y, \theta)^T$ be the coordinates that define the base position and orientation in a plane. Also, let v and ω be the instantaneous linear and angular velocity commands respectively. The kinematic model can then be written as

$$\begin{pmatrix} \dot{x} & \dot{y} & \dot{\theta} \end{pmatrix} = \begin{pmatrix} v\cos\theta & v\sin\theta & \omega \end{pmatrix} \tag{5.47}$$

from which the nonholonomic constraint $\dot{y}\cos\theta - \dot{x}\sin\theta = 0$ can be derived. Essentially, it implies that movement is not allowed in the wheels' axis direction.

In order to reproduce the walking motion, the robot's base must follow, with the same orientation, the path followed by the projection of the demonstrator's pelvis onto the floor. This is an inverse kinematics problem i.e., find the velocity commands that allow the robot to reach a given pose. Common path planning frameworks address this problem. However, they are not suitable for cases where the goal is constantly changing, which is the case of human walking. They usually involve complex calculations that cannot be solved faster than the goal-changing rate, which makes the robot remain in a planning state constantly.

We propose Algorithm 3 to solve these issues. When initialized, it assumes the person's and the robot's footprint frames are coincident in an arbitrarily fixed reference frame. Then the relative transform between the person and the robot footprint is determined at each time step. When the robot is further than a certain margin to the reference, angular velocity commands orientate the robot towards the goal position. If the robot position is close enough, angular velocity commands align the robot with the goal orientation. Additionally, the possibility of moving backward is also considered.

Algorithm 3: Differential Drive Base Motion Transfer

/* $\epsilon, \delta, \lambda, \sigma$: design parameters */
/* $()_{yaw}$: yaw component of rotation */
/* $()_{x,y}$: x, y component of translation */
/* x axis is assumed as forward */

$\boldsymbol{T}_{rf}^{pf} = \boldsymbol{I}$;
$\boldsymbol{T}_{ro}^{po} = \boldsymbol{T}_{ro}^{rf} \boldsymbol{T}_{pf}^{po}$;
while *True* **do**
  $\boldsymbol{T}_{rf}^{pf} = \left(\boldsymbol{T}_{ro}^{pf}\right)^{-1} \boldsymbol{T}_{ro}^{po} \boldsymbol{T}_{po}^{pf}$; // Relative transform
  if $\left\| \boldsymbol{p}_{rf}^{pf} \right\| < \epsilon$ **then**
    $v = 0 \qquad \omega = \lambda \cdot \left(\boldsymbol{R}_{rf}^{pf}\right)_{yaw}$
  else if $\left(\boldsymbol{p}_{rf}^{pf}\right)_x < 0$ ***and*** $\left| \left(\boldsymbol{R}_{rf}^{pf}\right)_{yaw} \right| < \delta$ **then**
    $v = -\sigma \cdot \left\| \boldsymbol{p}_{rf}^{pf} \right\|$
    $\omega = \lambda \cdot \left(\arctan \frac{\left(\boldsymbol{p}_{rf}^{pf}\right)_y}{\left(\boldsymbol{p}_{rf}^{pf}\right)_x} - \pi \cdot \text{sign} \left[\arctan \frac{\left(\boldsymbol{p}_{rf}^{pf}\right)_y}{\left(\boldsymbol{p}_{rf}^{pf}\right)_x} \right] \right)$
  else
    $v = \sigma \cdot \left\| \boldsymbol{p}_{rf}^{pf} \right\| \qquad \omega = \lambda \cdot \arctan \frac{\left(\boldsymbol{p}_{rf}^{pf}\right)_y}{\left(\boldsymbol{p}_{rf}^{pf}\right)_x}$
  end
end

5.4.5 Experiments

We carried out a series of real-world experiments using the TIAGo robot and the Xsens MVN motion capture system to validate our whole-body human-to-robot motion transfer framework. These can be divided into three different groups according to the aspect in which we focus the experimental analysis.

In the first group of experiments, we evaluate the similarity between the demonstrator's and the robot's upper body motion. The objective is to verify the correctness of the proposed solution for the correspondence problem and the tracking precision of the hierarchical WBC. In the second group, we analyze the behavior of the proposed role adaptive admittance controller scheme. The TIAGo robot must be able to respond compliantly to external forces on the end-effector, preserve a good tracking precision in their absence, and guarantee a stable behavior. Finally, in the third group of experiments, we study the performance of our algorithm for imitating the walking motion with the differential drive mobile base.

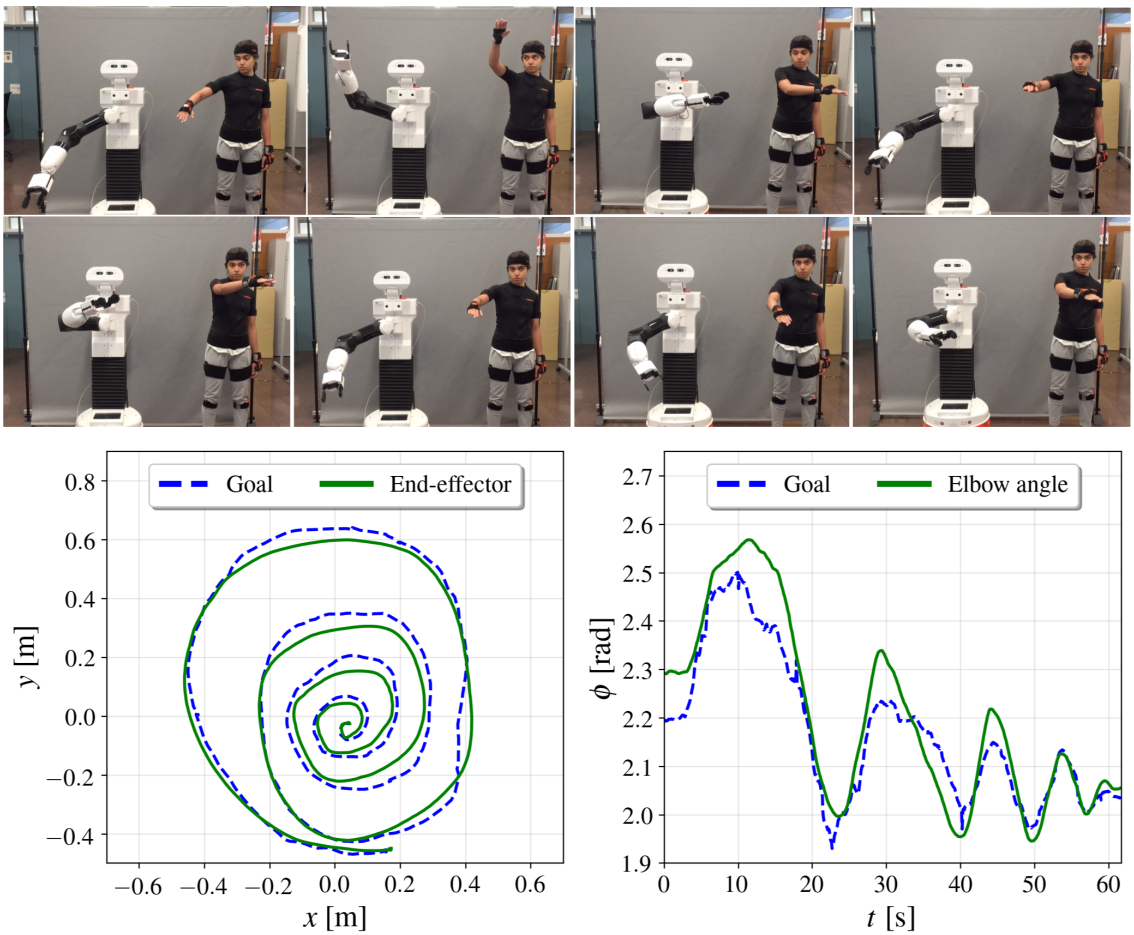

Figure 5.17: The demonstrator performs a spiral trajectory with the hand while the TIAGo robot imitates the upper body motion. On the lower-left the robot's end-effector trajectory against the goal trajectory. On the lower-right the evolution over time of the person's and the robot's angle ϕ between the elbow-wrist and elbow-shoulder segments.

Upper Body Motion Transfer

For evaluating the upper body motion transfer system, the robot performs real-time imitation while the demonstrator describes a spiral trajectory with the hand. We compare the trajectory followed by the robot's end-effector and the evolution of the angle formed by its elbow-wrist and elbow-shoulder segments with the demonstrator's reference. The results are shown in Figure 5.17. We can observe that the robot describes the spiral with the end-effector accurately while imitating the arm posture. This indicates, on the one hand, that the proposed solution of the correspondence problem allows the reproduction of the desired behavior with TIAGo using the demonstrator's motion. On the other hand, the whole-body controller effectively handles the redundant robot's DoF, achieving an online accurate reference tracking with multiple links of the robot simultaneously. This is also verified qualitatively in further experiments, depicted in Figure 5.18.

Figure 5.18: Several demonstrations of upper body motion transfer system performance. The robot accurately reproduces the movements of the human teacher. As we can see in some of the experiments depicted in this figure, our system opens the door for non-robotic experts to teach manipulation skills to a robot intuitively.

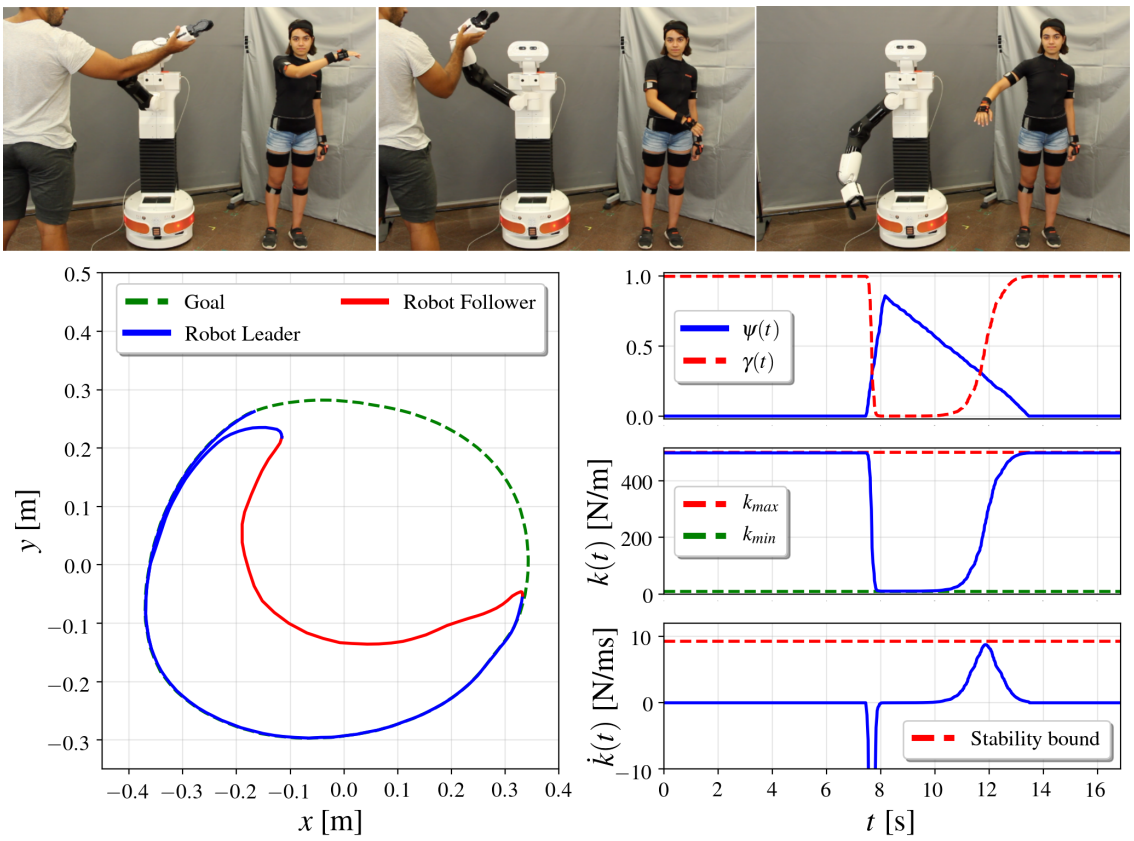

Figure 5.19: The demonstrator performs a circular trajectory with the hand while the TIAGo robot reproduces the motion. During the experiment, physical human-robot interaction occurs when a person grabs the end-effector. The TIAGo robot rapidly switches from the leader to the follower role, reducing its stiffness and yielding compliantly to the external force. When the end-effector is released the TIAGo robot switches back to the leader role in a stable manner, reproducing again the demonstrator's motion.

Role Adaptative Admittance Control

In order to study the behavior of the role adaptive variable admittance controller, we perform an experiment where TIAGo's end-effector is grasped while reproducing the demonstrator's motion and then released after a few seconds. The controller parameters are tuned empirically, selecting $\zeta = 1.1$, $k_{min} = 10$, $k_{max} = 500$, $c_1 = 20$, $c_2 = 0.275$, $c_3 = 1.5$ and $c_4 = 0.2$ for all six end-effector DoF. The experiment results are shown in Figure 5.19. We can see that when the grasping occurs, the interaction factor $\psi(t)$ starts to increase, while the stiffness $k(t)$ rapidly decreases to switch the robot behavior from stiff to compliant. This allows to easily move the end-effector away from its commanded trajectory. When it is released, the interaction factor starts to decrease while the stiffness starts to restore its initial value and the robot's end-effector position converges again to the reference trajectory. No oscillations or unstable behavior appeared during the experiment.

Figure 5.20: Several demonstrations of the mobile base motion transfer system performance with TIAGo. The robot accurately reproduces the walking motion of the human demonstrator. This includes forward and backward motions, and rotations. In some of the experiments, upper body motion is also transferred simultaneously.

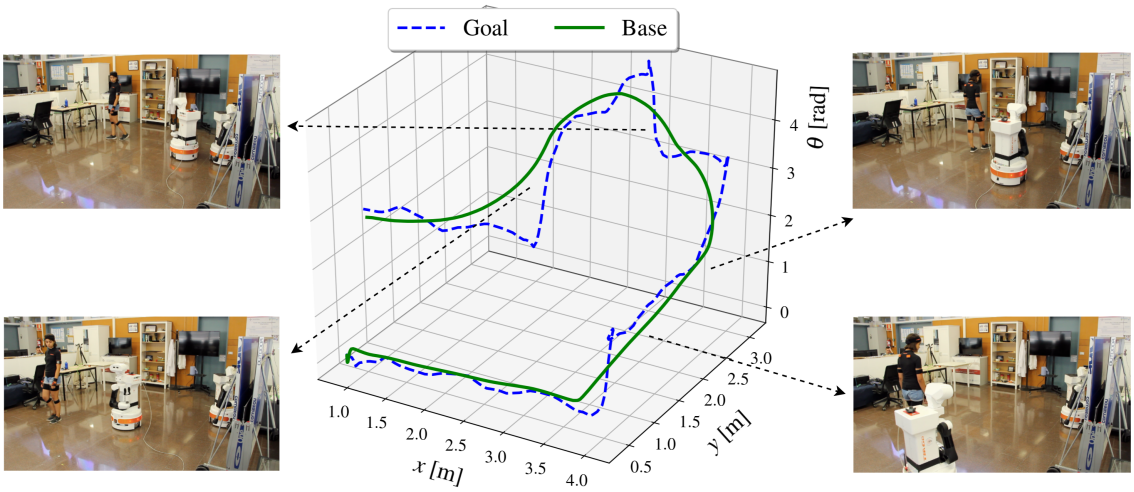

Figure 5.21: The demonstrator walks and describes a series of turns while the TIAGo robot imitates the motion with its mobile base. The plot shows the position (x and y) and orientation (θ) along with the path described by the robot's base (continuous line) compared with the reference trajectory provided by the demonstrator (dashed line).

Base Motion Transfer

Several qualitative tests of the mobile base motion transfer system are depicted in Figure 5.20. Additionally, we performed a more quantitative experiment where the demonstrator describes a series of turns while the TIAGo robot is reproducing the motion online, moving in parallel with the demonstrator. The results are shown in Figure 5.21. We can clearly observe that both, the robot position and orientation are very similar to the reference trajectory. Therefore, we can conclude that the proposed algorithm for differential drive base control allows the TIAGo robot to imitate the demonstrator's walking motion through velocity commands. It should be remarked that the non-holonomic constraint does not apply to human walking motion. Then, due to the difference in affordances, in order to achieve a successful imitation, the demonstrator trajectory should not include side steps.

Chapter 6

Learning from Demonstration with Gaussian Processes

Learning from demonstration (LfD) is the paradigm in which robots implicitly learn task constraints and requirements from demonstrations of a human teacher. This allows more intuitive skill transfer, satisfying a need of opening policy development to non-robotic-experts as robots extend to assistive domains. Flexible models that allow learning the task by extracting relevant motion patterns from the demonstrations, and subsequently apply these patterns to perform the task in different situations, are essential for transferring human skills to robots.

For addressing the learning from demonstration problem, we can assume that there exists a direct and learnable function (i.e., the policy) that generates the desired behavior. This policy can be defined as a function that maps available information onto an appropriate action space

$$\pi : \mathcal{X} \longrightarrow \mathcal{Y} \tag{6.1}$$

where $\mathcal{X}$ is the domain (the input space of the policy) and $\mathcal{Y}$ is its codomain (action space). The objective is to learn this policy $\pi()$, which allows the reproduction of the skill taught by the expert. For this, the robot is presented with a demonstration (i.e. training) dataset which consists of sample input-action pairs

$$\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N} = (X, Y) \tag{6.2}$$

where $\boldsymbol{x}_i \in \mathcal{X}$, $\boldsymbol{y}_i \in \mathcal{Y}$, N stands for the number of samples, and $X \in \mathbb{R}^{\dim(\mathcal{X})\times N}$ and $Y \in \mathbb{R}^{\dim(\mathcal{Y})\times N}$ represent the matrices where all the column input and output vectors are aggregated, respectively. Note that this statement is analogous to the supervised learning problem, tackled with Gaussian Processes in Chapter 3.

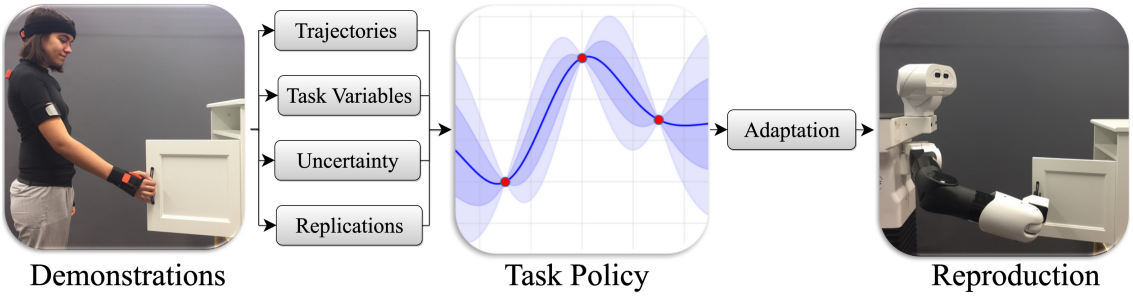

Figure 6.1: Gaussian-Process-based learning from demonstration framework overview.

From the formulation of the problem, we can see that the first key aspect in LfD involves identifying the appropriate inputs and outputs to the policy. Regarding the latter, trajectories are the most popular choice since in a myriad of robotic systems these govern the robot actions. On the other hand, the input captures the information that is necessary for generating the optimal actions, playing a central role in the generalization capabilities of the learned policy.

Another fundamental feature of LfD methods is the possibility of retrieving a probabilistic representation of the policy. This allows a complete description of the task, encoding the uncertainty along with the motion; which is crucial for reflecting the importance of certain points of the task, leading to better generalization capabilities. Also, in LfD it is interesting to adapt the learned motion to unseen scenarios while maintaining the general trajectory shape as in the demonstrations without re-training the model. Commonly, these requirements are expressed as via-point constraints or the blending of multiple movement policies. Finally, one should keep in mind that providing enough demonstrations to cover the input and action spaces is essential. When the model is probabilistic, repetitions are also required for capturing adequately the statistics of the taught motion. Thus, for learning complex manipulation tasks the amount of required data might increase considerably.

Taking advantage of the versatility and expressiveness of Gaussian Processes (GPs), in this chapter we aim to unify in a single framework, entirely GP-based, the main features required for state-of-the-art learning from demonstration method. An overview is depicted in Figure 6.1. We start in Section 6.1 by discussing how to learn trajectories with Gaussian Processes. Then, in Section 6.2, we show how to model the task uncertainty in general, and for time-invariant policies. Afterward, in Section 6.3, we introduce some kernels for including integer and categorical, as well as real input variables, in the model. Next, in Section 6.4, we discuss the adaptation of the learned policy, through via-point constraints and combination with other policies. Thereafter, in Section 6.5, we present an approach for exploiting the structure of replications for alleviating the computational complexity of Gaussian Processes. Finally, in Section 6.6, we illustrate the main ideas discussed throughout the chapter by means of a series of real-world experiments.

6.1 Learning Trajectories

The first fundamental question to design methods for learning robot manipulation tasks from demonstrations is, what should be learned? We have to decide which are the input space $\mathcal{X}$ and the action space $\mathcal{Y}$ that the learned policy $\pi()$ maps. The learning outcome depends on the level of abstraction that is appropriate, and thus chosen, for the problem of interest [9].

In robot learning from demonstration, the most popular approach is to select the space of trajectories as the action space, since this allows a direct skill transfer from human motion to robot actions. Learning a skill at a trajectory level involves encoding trajectories of certain variables of interest by extracting patterns from the available demonstrations. Examples of such variables include end-effector position, end-effector pose, and joint state. This approach has proven to be particularly well suited for over actuated systems, such as redundant manipulators, for which kinematic feasibility is relatively easier to achieve. Another important advantage is that there is no need for a model of the robot dynamics.

In this section, we present some extensions of the classical Gaussian Process formulation for modeling trajectory-based policies. First, in Section 6.1.1, we show how to adapt the single-output GP for learning functions with a multi-dimensional output space. Then, in Section 6.1.2, we discuss how to incorporate derivative information such as velocities or accelerations for making predictions and inference with Gaussian Processes. Finally, in Section 6.1.3, we introduce some covariance functions for considering rigid-body motions with rotations in the input space.

6.1.1 Multi-output Gaussian Processes

Commonly, the trajectories required to perform robot manipulation tasks are encoded by multiple coordinates. However, the standard Gaussian Process formulation focuses on the problem of predicting a single output variable from an input $\boldsymbol{x}$, which can be multi-dimensional. Gaussian Processes handle multi-dimensional input spaces naturally thanks to the kernel treatment. However, considering more than one output variable is more challenging.

To introduce the notation, we start by recalling some of the fundamental concepts of single-output Gaussian Processes. Essentially, a GP is defined by its prior scalar mean $m(\boldsymbol{x})$ and covariance $k(\boldsymbol{x}, \boldsymbol{x}')$ functions

$$\pi(\boldsymbol{x}) \sim \mathcal{GP}\left(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')\right) \tag{6.3}$$

Then, given the demonstration dataset $\mathcal{D} = (X, \boldsymbol{y})$ we are interested in incorporating the knowledge that it provides about the task policy $\pi()$. This operation corresponds to condition the joint Gaussian prior distribution on the observations

$$\pi(X_*) \sim \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \tag{6.4}$$
$$\boldsymbol{\mu}_* = m(X_*) + k(X_*, X)k(X, X)^{-1}(\boldsymbol{y} - m(X)) \tag{6.5}$$
$$\boldsymbol{\Sigma}_* = k(X_*, X_*) - k(X_*, X)k(X, X)^{-1}k(X, X_*) \tag{6.6}$$

where $\pi(X_*)$ are the actions predicted by the posterior task policy at the test input locations X_*, $m(X)$ denotes the vector resulting from evaluating the scalar mean function $m(\boldsymbol{x})$ at all the dataset's input locations X, and $k(X_*, X)$ the matrix resulting from evaluating the scalar covariance function for all input pairs $(\boldsymbol{x}, \boldsymbol{x}_*)$.

For multi-dimensional trajectories we need to predict the value of multiple variables simultaneously. This situation requires that the covariance function models not only the correlation structure of each output, but also the cross-correlations between them. We can obtain a multi-output Gaussian Process (MOGP) replacing the scalar mean $m(\boldsymbol{x})$ and covariance $k(\boldsymbol{x}, \boldsymbol{x}')$ functions in (6.3) by their multi-dimensional counterparts

$$\boldsymbol{\pi}(\boldsymbol{x}) \sim \mathcal{GP}(\boldsymbol{m}(\boldsymbol{x}), \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}')) \tag{6.7}$$

For a m-dimensional output $\boldsymbol{\pi}(\boldsymbol{x}) \in \mathbb{R}^m$ the covariance $\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}')$ can be written as

$$\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}') = \begin{bmatrix} k_{11}(\boldsymbol{x}, \boldsymbol{x}') & \dots & k_{1m}(\boldsymbol{x}, \boldsymbol{x}') \\ \vdots & \ddots & \vdots \\ k_{m1}(\boldsymbol{x}, \boldsymbol{x}') & \dots & k_{mm}(\boldsymbol{x}, \boldsymbol{x}') \end{bmatrix} \tag{6.8}$$

The off-diagonal elements $k_{ij}(\boldsymbol{x}, \boldsymbol{x}')$ with $i \neq j$, represent the prior covariance between two different outputs $\pi_i(\boldsymbol{x})$ and $\pi_j(\boldsymbol{x})$. On the other hand, the interpretation of the diagonal elements $k_{ii}(\boldsymbol{x}, \boldsymbol{x}')$ is analogous to the single-output case. The predictive equations for the multi-output Gaussian Process are identical to (6.4), (6.5) and (6.6), but arranging the elements adequately. For a m-dimensional output we would have a posterior distribution of the form

$$\boldsymbol{\pi}(X_*) = \begin{bmatrix} \pi_1(X_*) \\ \vdots \\ \pi_m(X_*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{1*} \\ \vdots \\ \boldsymbol{\mu}_{m*} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{11*} & \dots & \boldsymbol{\Sigma}_{1m*} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{m1*} & \dots & \boldsymbol{\Sigma}_{mm*} \end{bmatrix} \right) \tag{6.9}$$

Denoting by $\boldsymbol{k}(X, X_*)$ the covariance matrix constructed by evaluating all scalar functions $k_{ij}(,)$ at all possible input pairs

$$\boldsymbol{k}(X, X_*) = \begin{bmatrix} k_{11}(X, X_*) & \dots & k_{1m}(X, X_*) \\ \vdots & \ddots & \vdots \\ k_{m1}(X, X_*) & \dots & k_{mm}(X, X_*) \end{bmatrix} \tag{6.10}$$

and analogously for $\boldsymbol{k}(X, X)$, $\boldsymbol{k}(X_*, X)$ and $\boldsymbol{k}(X_*, X_*)$, the posterior mean and covariance matrices of equation (6.9) yield

$$\begin{bmatrix} \boldsymbol{\mu}_{1*} \\ \vdots \\ \boldsymbol{\mu}_{m*} \end{bmatrix} = \begin{bmatrix} m_1(X_*) \\ \vdots \\ m_m(X_*) \end{bmatrix} + \boldsymbol{k}(X_*, X)\boldsymbol{k}(X, X)^{-1} \left(\begin{bmatrix} Y_1 \\ \vdots \\ Y_m \end{bmatrix} - \begin{bmatrix} m_1(X_*) \\ \vdots \\ m_m(X_*) \end{bmatrix} \right) \tag{6.11}$$

$$\begin{bmatrix} \boldsymbol{\Sigma}_{11*} & \dots & \boldsymbol{\Sigma}_{1m*} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{m1*} & \dots & \boldsymbol{\Sigma}_{mm*} \end{bmatrix} = \boldsymbol{k}(X_*, X_*) - \boldsymbol{k}(X_*, X)\boldsymbol{k}(X, X)^{-1}\boldsymbol{k}(X, X_*) \tag{6.12}$$

where Y_j corresponds to the vector that contains all the observations of the j-th output dimension. When modeling trajectories we might don't know anything in advance about how the outputs relate to each other. Because of that, a common simplificative assumption is that $k_{ij}(\boldsymbol{x}, \boldsymbol{x}') = 0$, for $i \neq j$. The covariance matrix hence becomes diagonal

$$\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}') = \begin{bmatrix} k_{11}(\boldsymbol{x}, \boldsymbol{x}') & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & k_{mm}(\boldsymbol{x}, \boldsymbol{x}') \end{bmatrix} \tag{6.13}$$

By inspection of (6.10), (6.11) and (6.12), we can see that in this case the multi-output Gaussian Process with the covariance matrix (6.13) is equivalent to m single-output Gaussian Processes. For a more general case, approaches for modelling vector-valued functions with Gaussian Processes are mostly formulated around the linear model of coregionalization (LMC) [78]. Covariance functions obtained under the LMC assumption follow the form of a sum of Q separable kernels

$$\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{q=1}^{Q} \boldsymbol{B}_q k_q(\boldsymbol{x}, \boldsymbol{x}') \tag{6.14}$$

where each $\boldsymbol{B}_q \in \mathbb{R}^{m \times m}$ is a symmetric and positive semi-definite matrix known as the coregionalization matrix.

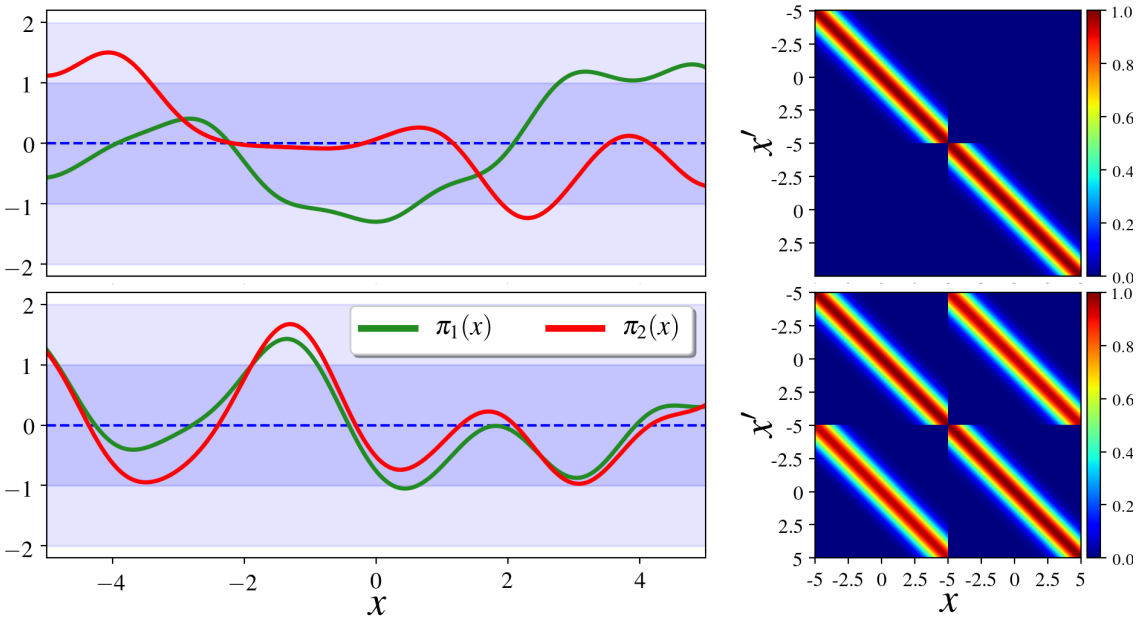

Figure 6.2: On top, samples drawn for a zero-mean multi-output Gaussian Process with a diagonal covariance function, shown on the right. Below, samples drawn for a zero-mean Gaussian Process formulated around the linear model of coregionalization. The prior scalar covariance for both cases is a SE kernel (6.16) with parameters $\ell = 1$ and $\sigma_f = 1$.

Equation (6.14) can be interpreted as the sum of the products of two covariance functions. One that models the dependence between the outputs, independently of the input vector $\boldsymbol{x}$ (the coregionalization matrix $\boldsymbol{B}_q$), and one that models the input dependence, independently of the particular set of output functions (the covariance function $k_q(\boldsymbol{x}, \boldsymbol{x}')$).

For the particular case of $Q = 1$, the computations required for inference are simplified significantly, since the prior distribution can be expressed just as

$$\boldsymbol{\pi}(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{m}(X), \boldsymbol{B} \otimes k(X, X)) \tag{6.15}$$

where $\otimes$ denotes the Kronecker product. An illustrative example of a comparison between a multi-output Gaussian Process considering all output dimensions as independent, and another formulated around the LMC is shown in Figure 6.2. As the scalar covariance function, we take the squared-exponential (SE) kernel

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{1}{2}\frac{(x - x')^2}{\ell^2}\right) \tag{6.16}$$

We can observe that the samples drawn from the prior for each output are quite different considering a diagonal coregionalization matrix $\boldsymbol{B}$. On the other hand, including positive off-diagonal elements in the coregionalization matrix $\boldsymbol{B}$, we are assuming that for similar input values both outputs $\pi_1(x)$ and $\pi_2(x)$ should have a similar value. As we can see, the samples drawn from the posterior in this case are very similar to each other. Thus, for modeling trajectories for which we have some prior knowledge about the relation among the different output dimensions, an adequate multi-output design is essential to express this structure in the policy.

6.1.2 Derivative of a Gaussian Process

In the representation of trajectories, they usually appear derivative relations e.g. velocities and accelerations. Since differentiation is a linear operator, the derivative of a Gaussian Process is also a Gaussian Process. Thus we can use GPs to make predictions about derivatives, and also make inferences based on derivative information [79]. For developing this concept, consider the following general prior distribution for a two-dimensional output Gaussian Process

$$\begin{bmatrix} y(\boldsymbol{x}) \\ y_d(\boldsymbol{x}) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\boldsymbol{x}) \\ m_d(\boldsymbol{x}) \end{bmatrix}, \begin{bmatrix} k(\boldsymbol{x},\boldsymbol{x}') & k_d(\boldsymbol{x},\boldsymbol{x}') \\ k_d(\boldsymbol{x}',\boldsymbol{x}) & k_{dd}(\boldsymbol{x},\boldsymbol{x}') \end{bmatrix}\right) \tag{6.17}$$

We know that the output $y_d(\boldsymbol{x})$ corresponds to the derivative of $y(\boldsymbol{x})$

$$y_d(\boldsymbol{x}) = \frac{dy(\boldsymbol{x})}{d\boldsymbol{x}} \tag{6.18}$$

How can we incorporate this knowledge into the multi-output Gaussian Process? First, we can relate the prior mean functions substituting (6.18) in the definition of $m_d(\boldsymbol{x})$. Taking advantage that both, the expectation and the derivative operators are linear operators we can hence change the order in which they are applied

$$m_d(\boldsymbol{x}) = \mathbb{E}\left[y_d(\boldsymbol{x})\right] = \mathbb{E}\left[\frac{dy(\boldsymbol{x})}{d\boldsymbol{x}}\right] = \frac{d}{d\boldsymbol{x}}\mathbb{E}\left[y(\boldsymbol{x})\right] = \frac{dm(\boldsymbol{x})}{d\boldsymbol{x}} \tag{6.19}$$

This means that to find the mean of the derivative of a Gaussian Process, we can just take the derivative of the mean. Applying the same trick we can also compute how are the diagonal terms of the covariance matrix (6.17), related to each other

$$k_{dd}(\boldsymbol{x},\boldsymbol{x}') = \mathbb{E}\left[\left(\frac{dy(\boldsymbol{x})}{d\boldsymbol{x}} - \frac{dm(\boldsymbol{x})}{d\boldsymbol{x}}\right)\left(\frac{dy(\boldsymbol{x}')}{d\boldsymbol{x}'} - \frac{dm(\boldsymbol{x}')}{d\boldsymbol{x}'}\right)\right] = \frac{d^2k(\boldsymbol{x},\boldsymbol{x}')}{d\boldsymbol{x}d\boldsymbol{x}'} \tag{6.20}$$

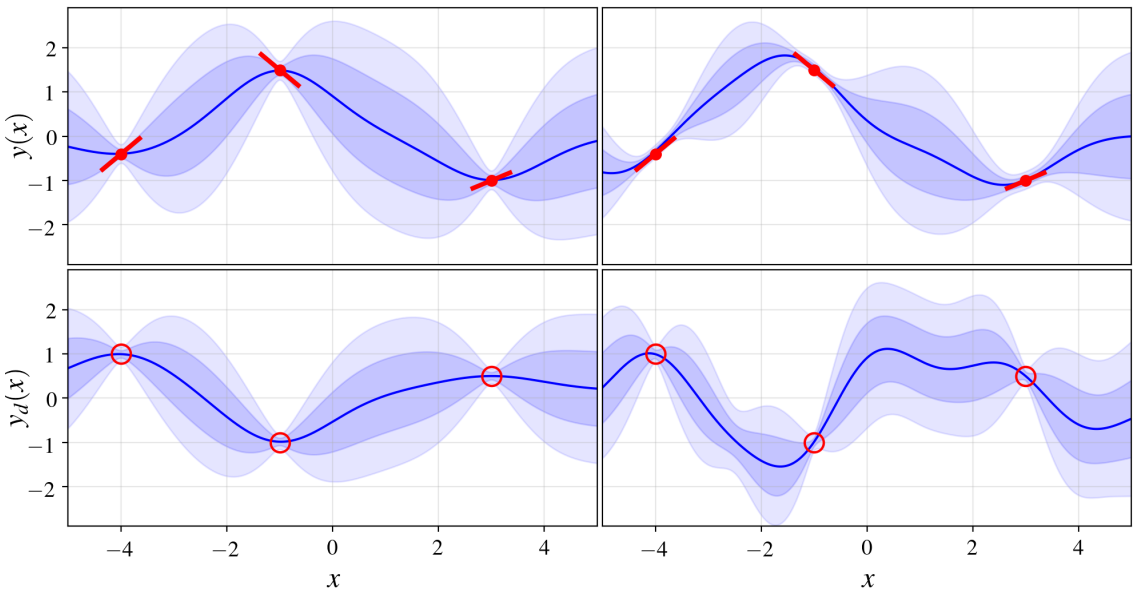

Figure 6.3: On the left column, two independent zero-mean Gaussian Processes for $y(x)$ and its derivative $y_d(x)$, with prior covariance function (6.23) and $(\sigma_f, \ell, \sigma_n) = (1, 1, 0.01)$. We use three training points for each output, marked as red dots. The derivative measurements are also represented in $y(x)$ as red stripes. On the right column, multi-output Gaussian Process taking into account the derivative relation between the outputs (6.21).

Finally, for the off-diagonal terms $k_d(\boldsymbol{x}, \boldsymbol{x}')$ we have

$$k_d(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}\left[(y(\boldsymbol{x}) - m(\boldsymbol{x}))\left(\frac{dy(\boldsymbol{x}')}{d\boldsymbol{x}'} - \frac{dm(\boldsymbol{x}')}{d\boldsymbol{x}'}\right)\right] = \frac{dk(\boldsymbol{x}, \boldsymbol{x}')}{d\boldsymbol{x}'} \tag{6.21}$$

Substituting (6.19), (6.20) and (6.21) in (6.17), the joint prior distribution considering the differential relation between the outputs yields

$$\begin{bmatrix} y(\boldsymbol{x}) \\ y_d(\boldsymbol{x}) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\boldsymbol{x}) \\ \frac{dm(\boldsymbol{x})}{d\boldsymbol{x}} \end{bmatrix}, \begin{bmatrix} k(\boldsymbol{x}, \boldsymbol{x}') & \frac{dk(\boldsymbol{x}, \boldsymbol{x}')}{d\boldsymbol{x}'} \\ \frac{dk(\boldsymbol{x}', \boldsymbol{x})}{d\boldsymbol{x}} & \frac{d^2k(\boldsymbol{x}', \boldsymbol{x})}{d\boldsymbol{x}d\boldsymbol{x}'} \end{bmatrix}\right) \tag{6.22}$$

Then, prediction can be performed using (6.11) and (6.12). An example for illustrating the advantage of embedding the differential relation in the multi-output Gaussian Process design is shown in Figure 6.2. We assume a zero-mean prior and a SE kernel with additive white noise

$$k(x_i, x_j) = \sigma_f^2 \exp\left(-\frac{1}{2}\frac{(x_i - x_j)^2}{\ell^2}\right) + \sigma_n^2 \delta_{ij} \tag{6.23}$$

where δ_{ij} is the Kronecker delta. The derivatives of this kernel are the following

$$\frac{dk(x_i, x_j)}{dx_j} = \frac{dk(x_j, x_i)}{dx_i} = \frac{\sigma_f^2}{\ell^2}(x_i - x_j)\exp\left(-\frac{1}{2}\frac{(x_i - x_j)^2}{\ell^2}\right) \tag{6.24}$$

$$\frac{dk(x_i, x_j)}{dx_i dx_j} = \frac{\sigma_f^2}{\ell^4}\left(\ell^2 - (x_i - x_j)^2\right)\exp\left(-\frac{1}{2}\frac{(x_i - x_j)^2}{\ell^2}\right) \tag{6.25}$$

We can observe that considering the differential relation between the outputs, the resulting posterior for $y(x)$ takes also into account the derivative measurements, adjusting the slope of the posterior. On the contrary, considering both outputs as independent, the resulting model is not coherent with the differential relation. This gives an intuitive insight into the effect of considering observations of the function and its derivatives for computing the Gaussian Process posterior.

6.1.3 Gaussian Processes over Rigid-body Motions

The kernel functions are considered the core of GP modeling. However, most of the kernels are non-trivial to extend beyond the Euclidean space in general. When learning trajectories, the input space of the policy $\mathcal{X}$ might include rigid-body motions, which consist of the composition of a translation and a rotation. As for rotations, there exists no representation in Euclidean space, they cannot be modeled accurately with the classical GP formulation. In this section, we discuss how to generalize the common kernels to consider the special Euclidean group $SE(3)$ of 6 degree-of-freedom rigid body motions in the input space [80]. For this purpose, we first address the representation of rotations, then the definition of a distance metric in the $SE(3)$ manifold, and finally the construction of suitable kernel functions.

Axis-angle and Translation Vector Representation

From Euler's fixed point theorem [81], we know that in 3D space, any rigid-body displacement with one fixed point, i.e. any composition of rotations, can be equivalently described by a single rotation about some non-trivial axis of rotation through the fixed point (Figure 6.4). This axis is also called the Euler axis. Thus, the set of unit length Euler axes $\boldsymbol{u}$ together with a rotation angle θ parametrizes the rotation group, called special orthogonal group $SO(3)$

$$SO(3) \subset \left\{\theta\boldsymbol{u} \in \mathbb{R}^3 \mid \|\boldsymbol{u}\| = 1 \wedge \theta \in [0, \pi]\right\} \tag{6.26}$$

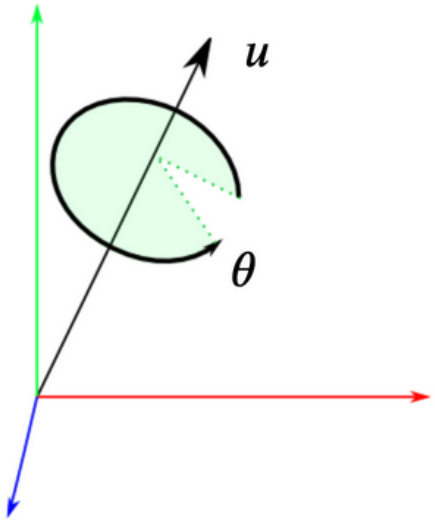

Figure 6.4: Axis ($\boldsymbol{u}$) - angle (θ) representation of rotations.

where the symbol $\wedge$ denotes the logical 'and'. The set (6.26) defines the solid ball $B_\pi(0)$ in $\mathbb{R}^3$ with radius $0 \leq r \leq \pi$ centered around the origin and is thus closed, dense and compact. Ambiguity in the representation occurs for $\theta = \pi$, as $\pi\boldsymbol{u} = -\pi\boldsymbol{u}$ define the same rotation. To obtain a homeomorphism (i.e. a one to one relation) between the rotation group $SO(3)$ and the axis-angle representation, we additionally fix the Euler axis representation for $\theta = \pi$ and obtain

$$\widetilde{B}_\pi(0) = B_\pi(0) \setminus \{\pi\boldsymbol{u} \mid u_z < 0 \vee (u_z \wedge u_y < 0) \vee (u_z = u_y = 0 \wedge u_x < 0)\} \quad (6.27)$$

where the symbol $\vee$ denotes the logical 'or', $\setminus$ refers to the subtraction operation for sets, and u_x, u_y and u_z are the components of the axis vector $\boldsymbol{u}$. This parametrization of the rotation group by an Euler axis and a rotation angle is minimal and unique, $SO(3) \simeq \widetilde{B}_\pi(0)$. Therefore, the link with translations $\boldsymbol{v} \in \mathbb{R}^3$ can be realized using the standard Cartesian set product. The spaces of rotation $\widetilde{B}_\pi(0)$ and translation $\mathbb{R}^3$ jointly define a homeomorphism to $SE(3)$

$$SE(3) \simeq \widetilde{B}_\pi(0) \times \mathbb{R}^3 \quad (6.28)$$

Thus, any 6 degree-of-freedom rigid-body motion $\boldsymbol{s} \in SE(3)$ can be represented by the pair $\boldsymbol{s} = (\theta\boldsymbol{u}, \boldsymbol{v}) \in \widetilde{B}_\pi(0) \times \mathbb{R}^3$. Additionally, the dot product is defined as in regular vector spaces

$$\langle \boldsymbol{s}_1, \boldsymbol{s}_2 \rangle = \langle \theta_1\boldsymbol{u}_1, \theta_2\boldsymbol{u}_2 \rangle + \langle \boldsymbol{v}_1, \boldsymbol{v}_2 \rangle = |\theta_1||\theta_2| \cos \sphericalangle \boldsymbol{u}_1\boldsymbol{u}_2 + \langle \boldsymbol{v}_1, \boldsymbol{v}_2 \rangle \quad (6.29)$$

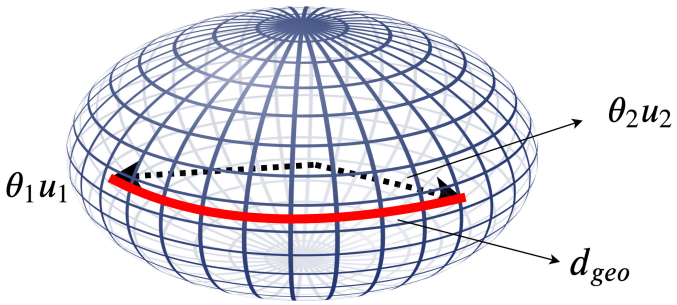

Figure 6.5: Geodesic distance d_{geo} between rotations for the axis-angle representation.

Distance Metrics

The prominent squared exponential kernel (6.16) and most other kernel functions depend on a distance metric between input samples. We can define the distance over rotations as the length of the geodesic between rotations. An illustration is shown in Figure 6.5. For the axis angle representation, this distance can be computed as

$$d_{geo}(\theta_1 \boldsymbol{u}_1, \theta_2 \boldsymbol{u}_2) = 2 \arccos \left| \cos \frac{\theta_1}{2} \cos \frac{\theta_2}{2} + \sin \frac{\theta_1}{2} \sin \frac{\theta_2}{2} \boldsymbol{u}_1^T \boldsymbol{u}_2 \right| \tag{6.30}$$

Note that $0 \leq d_{geo} \leq 2\pi$. Then, as distance function on $SE(3)$ parametrized by axis-angle and translation vector we can use the root over a sum of squares. Hence, we define a distance function for rigid motions

$$d_s(\boldsymbol{s}_1, \boldsymbol{s}_2) = \sqrt{\rho_1 \left(d_{geo}(\theta_1 \boldsymbol{u}_1, \theta_2 \boldsymbol{u}_2) \right)^2 + \rho_2 \|\boldsymbol{v}_1 - \boldsymbol{v}_2\|^2)} \tag{6.31}$$

where ρ_1 and ρ_2 are a convex combination of weights, $\sum_i \rho_i = 1$ where $\rho_i \geq 0$, for an application-dependent scaling between rotation and translation, allowing the incorporation of domain knowledge. Depending on the underlying rigid-body motion, we can weigh the contribution of rotation and translation on the similarity measure between poses. For instance, for the motion of a sphere, we can set $\rho_1 = 0$.

Covariance Functions for Rigid-Body Motions

The covariance function of the Gaussian Process solely depends on the input domain $\mathcal{X}$. For a single-output Gaussian Processes over rigid body motions on $SE(3)$ the covariance is of the form

$$k(\boldsymbol{s}, \boldsymbol{s}') : SE(3) \times SE(3) \longrightarrow \mathbb{R} \tag{6.32}$$

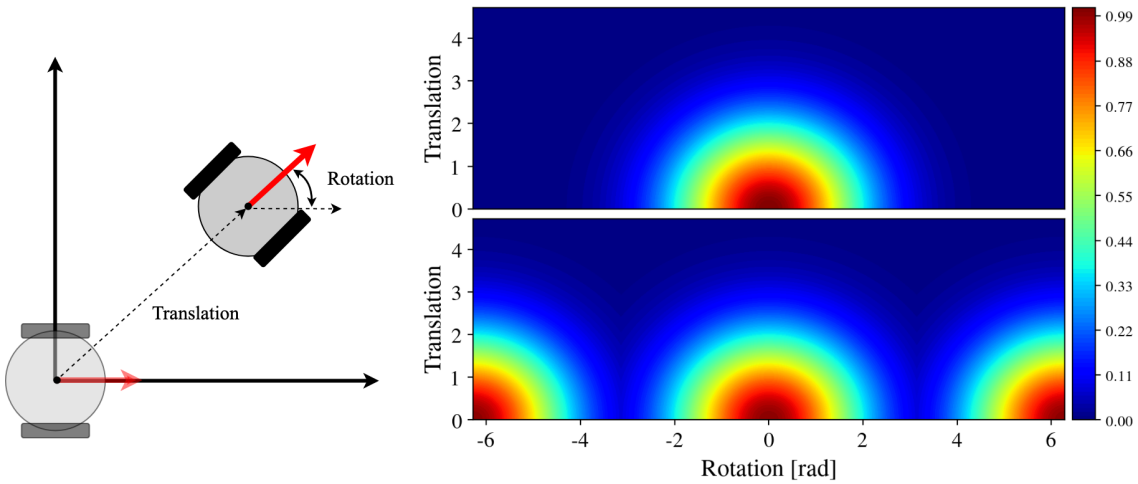

Figure 6.6: Comparison of stationary kernels over rigid-body motions. On the upper-right, squared-exponential kernel (6.23) evaluated for different values of the translation and rotation, computing the latter as Euclidean. Below, kernel (6.33), where the distance between rotations is computed as the geodesic of the axis-angle representation. The parameters are set to $\sigma_f = \ell = 1$, $\sigma_n = 0.01$ and $\rho_1 = \rho_2 = 0.5$.

This function must be symmetric from the definition, implying also that the Gram matrix must be positive semidefinite. The most common covariance functions can be divided into two main groups: stationary and dot product kernels. Essentially, a stationary covariance function is a function of $|\boldsymbol{x} - \boldsymbol{x}'|$. Thus it is invariant to translations in the input space. On the other hand, dot product covariance functions depend only on $\boldsymbol{x}$ and $\boldsymbol{x}'$ through $\boldsymbol{x} \cdot \boldsymbol{x}' \equiv \langle \boldsymbol{x}, \boldsymbol{x}' \rangle$. These are invariant to a rotation of the input space about the origin, but not translations.

Stationary Kernels: These type of kernels, commonly defined for an underlying Euclidean space, can be adapted for $SE(3)$ replacing the distance $|\boldsymbol{x} - \boldsymbol{x}'|$ by (6.31). For instance, the squared-exponential kernel (6.23) over rigid-body motions yields

$$k(\boldsymbol{s}_i, \boldsymbol{s}_j) = \sigma_f^2 \exp\left(-\frac{\rho_1}{2}\frac{d_{geo}\left(\theta_i \boldsymbol{u}_i, \theta_j \boldsymbol{u}_j\right)^2}{\ell^2}\right) \exp\left(-\frac{\rho_2}{2}\frac{\|\boldsymbol{v}_i - \boldsymbol{v}_j\|^2}{\ell^2}\right) + \sigma_n^2 \delta_{ij} \quad (6.33)$$

A comparison of kernel (6.33) against its Euclidean counterpart (6.23), evaluating the covariance between the poses of a mobile robot is shown in Figure 6.5. We can see that using the axis-angle representation for rotations, and evaluating the distance between configurations adequately, the kernel captures much better the similarity between poses, taking into account the non-euclidean topology of $SE(3)$. This can be observed in the periodicity of the covariance matrix for the rotation coordinate.

Dot Product Kernels: Analogously to the stationary kernels, these can be adapted to rigid-body motions using the definition (6.29) of the dot product. The advantage of the axis-angle representation is that the dot product operation in $SE(3)$ is analogous to the dot product in Euclidean space. In this case, the main difference lies in the representation of the rotation component. Using the axis-angle we take into account the periodicity on $SO(3)$.

6.2 Modeling the Task Uncertainty

Encoding the task policy with Gaussian Processes we have a stochastic representation. That is, instead of assuming that a singular optimal action exists for every situation, we assume that there is an underlying distribution of ideal behaviors. The great advantage of a stochastic policy is its ability to capture the inherent task uncertainty as well as generalize across multiple demonstrations. Using Gaussian Processes we assume that the policy is distributed according to a multivariate Gaussian distribution, being the task uncertainty encoded in the covariance matrix, and the generalized representation of the policy in the mean.

In this section, we focus on modeling effectively the task uncertainty from the demonstrations with Gaussian Processes, for what some careful design considerations are required. On the one hand, in a standard Gaussian Process, the uncertainty is presumed to be constant in those regions where training data is available. However, this is not the case for a general manipulation task. On the other hand, for a human teacher is very difficult to always perform the task at the same speed. This introduces a time variability that can lead to an erroneous estimate of the task uncertainty when learning time-independent policies.

First, in Section 6.2.1, we present heteroscedastic Gaussian Processes, which relax the constant uncertainty assumption, introducing an input-dependent uncertainty in the model. Then, in Section 6.2.2, we discuss how to correct the time variability introduced by the human teacher in the demonstrations with the Dynamic Time Warping algorithm, traditionally used for temporal alignment of time series.

6.2.1 Heteroscedastic Gaussian Processes

The standard Gaussian Process formulation assumes identically distributed white noise in the kernel design. This can be an important limitation for capturing the variability in the demonstrations, as this assumption yields a constant uncertainty profile in those regions where data is available. This is illustrated in the example shown in Figure 6.7.

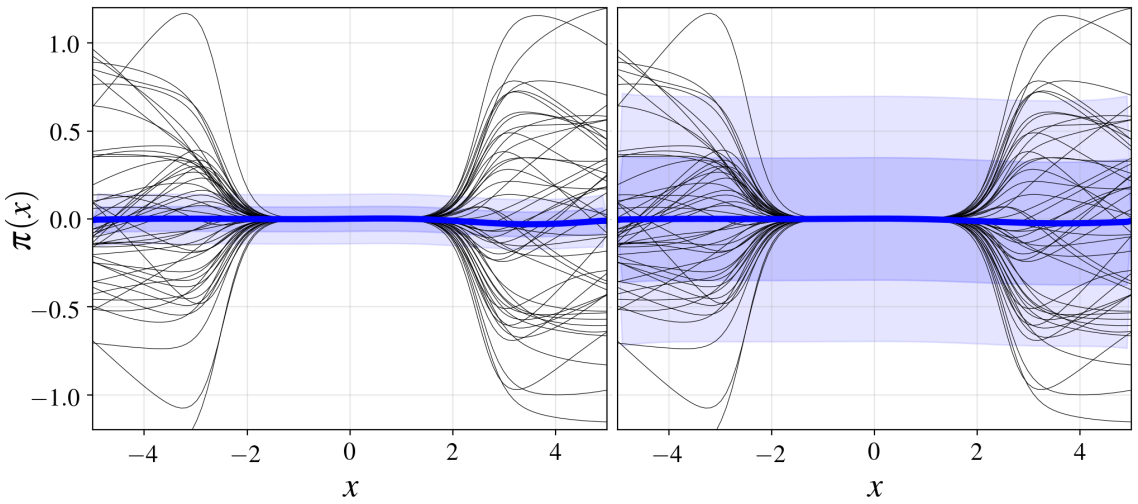

Figure 6.7: Task policy $\pi(x)$ (blue) inferred from the demonstrations (black) with a standard Gaussian Process. It assumes a constant uncertainty, which might lead to underestimate it in certain phases of the task (on the left) or overestimate it (on the right).

It is evident that while the intermediate positions are highly constrained, that is not the case for the starting and final positions. With the standard Gaussian Process model we are able to correctly estimate the mean, however, we totally fail to infer the inherent task uncertainty. Thus, we should relax the constant noise assumption to have a variable uncertainty profile. This kind of model is known in the literature as the heteroscedastic Gaussian Process [82].

In most of the cases, in robot learning from demonstration, we have that the uncertainty of the task policy is actually input-dependent. Thus, if y_i represents the demonstrated action for input $\boldsymbol{x}_i$, we can assume the following observation model

$$y_i = f(\boldsymbol{x}_i) + r(\boldsymbol{x}_i) \tag{6.34}$$

where $r(\boldsymbol{x}_i) \sim \mathcal{N}(0, k_r(\boldsymbol{x}_i, \boldsymbol{x}_j))$ is the latent task uncertainty function which we assume normally distributed and independent from $f(\boldsymbol{x}_i) \sim \mathcal{N}(m(\boldsymbol{x}_i), k(\boldsymbol{x}_i, \boldsymbol{x}_j))$, which is the function describing the underlying mean trajectory for performing the demonstrated task. Then, taking into account that the sum of two independent normally distributed random variables is normal, with its mean being the sum of the two means, and its variance being the sum of the two variances. Prior to providing any demonstration of the task, y_i is distributed according to

$$y_i \sim \mathcal{N}(m(\boldsymbol{x}_i), k(\boldsymbol{x}_i, \boldsymbol{x}_j) + k_r(\boldsymbol{x}_i, \boldsymbol{x}_j)) \tag{6.35}$$

Given the training dataset $\mathcal{D} = (X, \boldsymbol{y})$, we have that the demonstrated trajectories and the predicted mean trajectory for the new input locations X_* are jointly Gaussian distributed as

$$\begin{bmatrix} \boldsymbol{y} \\ f(X_*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(X) \\ m(X_*) \end{bmatrix}, \begin{bmatrix} k(X,X) + k_r(X,X) & k(X,X_*) \\ k(X_*,X) & k(X_*,X_*) \end{bmatrix} \right) \tag{6.36}$$

Then, the posterior distribution of the generalized trajectory $f(X_*)$ for performing the task conditioned on the demonstrations yields

$$f(X_*) \sim \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \tag{6.37}$$

$$\boldsymbol{\mu}_* = m(X_*) + k(X_*,X)\left(k(X,X) + k_r(X,X)\right)^{-1}(\boldsymbol{y} - m(X)) \tag{6.38}$$

$$\boldsymbol{\Sigma}_* = k(X_*,X_*) + k(X_*,X)\left(k(X,X) + k_r(X,X)\right)^{-1} k(X,X_*) \tag{6.39}$$

Until now, the development is identical to the predictive equations of the standard Gaussian Process with measurement error, but including an input-dependent noise. However, unlike the usual regression problem, note that we do not want to infer the underlying mean trajectory $f()$. This is the problem that we have in the example shown in Figure 6.7. Actually, we want our task policy $\pi()$ to encode jointly both, a generalized representation of the task $f()$ and its uncertainty $r()$. That is,

$$\pi(\boldsymbol{x}) = f(\boldsymbol{x}) + r(\boldsymbol{x}) \tag{6.40}$$

Therefore, we have that the posterior distribution of the actions retrieved by the policy conditioned on the demonstrations yields

$$\pi(X_*) \sim \mathcal{N}(\boldsymbol{\mu}_* + \mathbf{0}, \boldsymbol{\Sigma}_* + k_r(X_*,X_*)) = \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_{r*}) \tag{6.41}$$

$$\boldsymbol{\Sigma}_{r*} = k(X_*,X_*) + k_r(X_*,X_*) + k(X_*,X)\left(k(X,X) + k_r(X,X)\right)^{-1} k(X,X_*) \tag{6.42}$$

Intuitively, one can understand the term $\boldsymbol{\mu}_*$ as the mean trajectory, $\boldsymbol{\Sigma}_*$ as the uncertainty in the estimation of $\boldsymbol{\mu}_*$ and $k_r(X_*,X_*)$ as the inherent task uncertainty. This idea is illustrated in Figure 6.8. Since a large number of demonstrations are provided, $\boldsymbol{\Sigma}_*$ is almost null. By including the task uncertainty in the policy model we are able to capture the variability in the demonstrations.

Now the question is, how can we determine the covariance function $k_r(,)$ of the latent task uncertainty $r()$? For simplifying the problem, we can decouple the covariance as the product of the correlation and the variance

$$k_r(\boldsymbol{x}, \boldsymbol{x}') = \sigma_r(\boldsymbol{x})\sigma_r(\boldsymbol{x}')c(\boldsymbol{x}, \boldsymbol{x}') \tag{6.43}$$

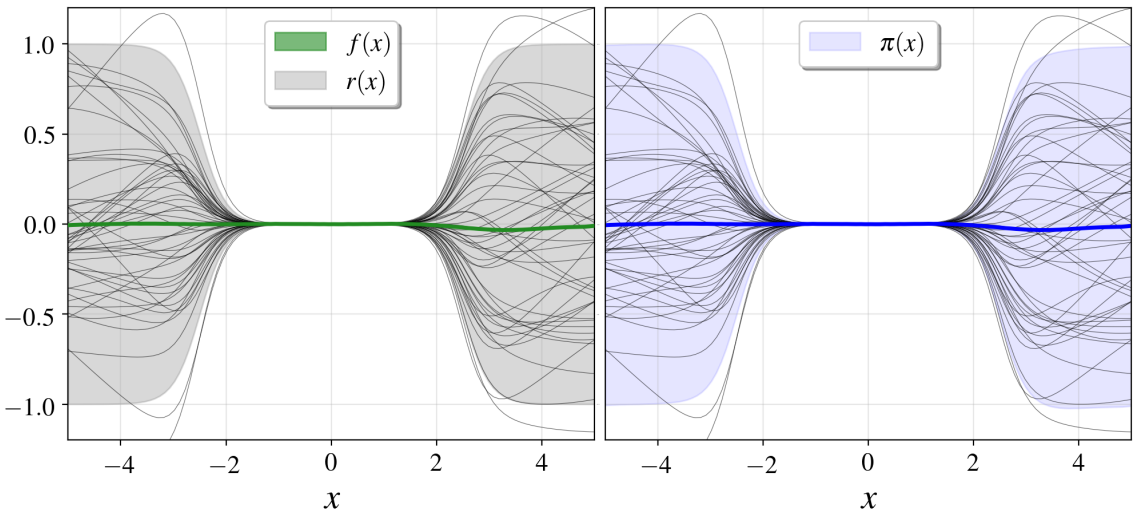

Figure 6.8: The task policy $\pi(x)$ encodes jointly the underlying mean trajectory $f(x)$ and the task uncertainty $r(x)$. The shaded areas represent two times the standard deviation and the solid lines the means. Due to the large amount of training data the uncertainty of $f(x)$ is negligible.

where $c(\boldsymbol{x}, \boldsymbol{x}')$ describes how is the task uncertainty at the input location $\boldsymbol{x}$ correlated with the uncertainty at the input location $\boldsymbol{x}'$, and $\sigma_r(\boldsymbol{x}')$ represents the magnitude of such uncertainty. The former is task-dependent and can be selected analogously to kernels in Gaussian Processes depending on the nature of the uncertainty. The heteroscedastic Gaussian Process literature always considers

$$c(\boldsymbol{x}_i, \boldsymbol{x}_j) = \delta_{ij} \tag{6.44}$$

However, this assumption is not adequate for robot manipulation tasks. For instance, consider the case shown in Figure 6.9. Assuming that the uncertainty is uncorrelated, the samples drawn from the posterior are very noisy and not suitable to be reproduced by a robot. Instead, by considering a squared exponential correlation for the noise

$$c(x, x') = \exp\left(-\frac{1}{2}\frac{(x - x')^2}{\ell^2}\right) \tag{6.45}$$

the samples drawn from the posterior are much more smooth. The intuitive idea behind this behavior is analogous to the standard Gaussian Process design. Considering a correlation of the form (6.44) we are assuming that the uncertainty in the task is explained by white noise, resulting in a very wiggly sample.

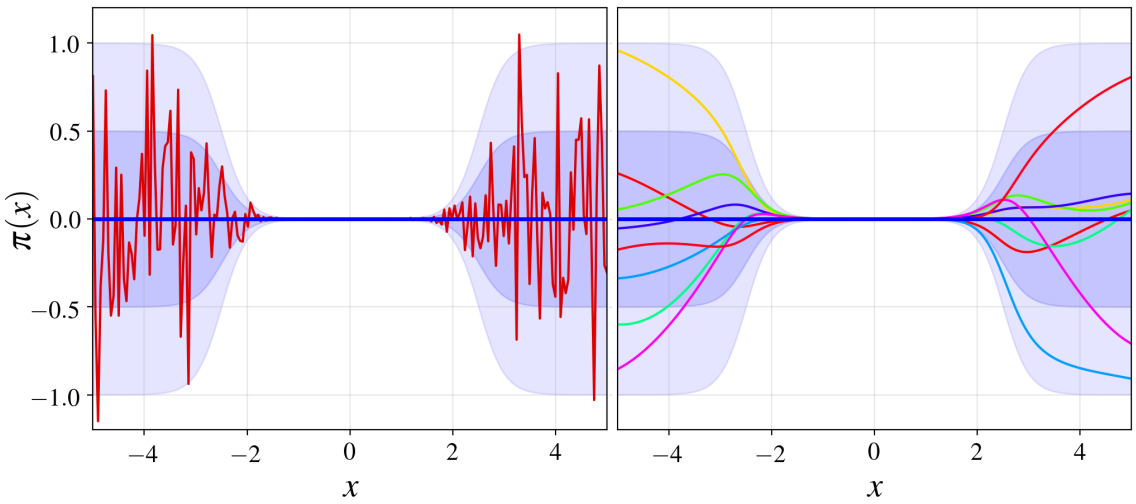

Figure 6.9: Comparison between samples drawn from the posterior predictive distribution considering uncorrelated noise (6.44) (on the left) and considering a correlation of the form (6.45) with $\ell = 2.5$ (on the right).

The input-dependent magnitude of the uncertainty $\sigma_r(\boldsymbol{x})$ can be inferred from the demonstrations. As proposed in [83], we can use an independent Gaussian Process to model the logarithm of the uncertainty, which we denote as

$$z(\boldsymbol{x}) = \log\left(\sigma_r^2(\boldsymbol{x})\right) \sim \mathcal{GP}\left(m_z(\boldsymbol{x}), k_z(\boldsymbol{x}, \boldsymbol{x}')\right) \tag{6.46}$$

By performing a regression of the logarithm, we ensure that the predicted variances are always positive after computing the exponential for recovering $\sigma_r^2(\boldsymbol{x})$. Now, let $\boldsymbol{z} = z(X)$ and $\boldsymbol{z}_* = z(X_*)$ be the (unknown) vectors obtained after evaluating $z(\boldsymbol{x})$ at the training and test input locations, respectively. Considering them as latent variables and marginalizing, we have that posterior predictive distribution of the task policy is given by

$$p\left(\pi(X_*) \mid \mathcal{D}, X_*\right) = \iint p\left(\pi(X_*) \mid \mathcal{D}, X_*, \boldsymbol{z}, \boldsymbol{z}_*\right) p(\boldsymbol{z}, \boldsymbol{z}_* \mid \mathcal{D}, X_*) d\boldsymbol{z} d\boldsymbol{z}_* \tag{6.47}$$

Given $\boldsymbol{z}$ and $\boldsymbol{z}_*$, the first term can be computed using equations (6.38) and (6.42). However, $p(\boldsymbol{z}, \boldsymbol{z}_* \mid \mathcal{D}, X_*)$ prevents an analytical solution. Thus, for computing the integral we can assume the following approximation

$$p\left(\pi(X_*) \mid \mathcal{D}, X_*\right) \simeq p(\pi(X_*) \mid \mathcal{D}, X_*, \widetilde{\boldsymbol{z}}, \widetilde{\boldsymbol{z}}_*) \tag{6.48}$$

where $\widetilde{\boldsymbol{z}}$ and $\widetilde{\boldsymbol{z}}_*$ are the most likely log-uncertainty levels

$$(\widetilde{\boldsymbol{z}}, \widetilde{\boldsymbol{z}}_*) = \arg\max_{\boldsymbol{z}, \boldsymbol{z}_*} p(\boldsymbol{z}, \boldsymbol{z}_* \mid \mathcal{D}, X_*) \tag{6.49}$$

This is a good approximation if most of the probability mass of $p(\boldsymbol{z}, \boldsymbol{z}_* \mid \mathcal{D}, X_*)$ is concentrated around the most likely values of $\boldsymbol{z}$ and $\boldsymbol{z}_*$. Considering the Gaussian Process model in (6.46) we have that they are given by

$$\widetilde{\boldsymbol{z}} = m_z(X) + k_z(X, X)k_z(X, X)^{-1}\left(\boldsymbol{z} - m_z(X)\right) \tag{6.50}$$
$$\widetilde{\boldsymbol{z}}_* = m_z(X_*) + k_z(X_*, X)k_z(X, X)^{-1}\left(\boldsymbol{z} - m_z(X)\right) \tag{6.51}$$

The problem is that we do not have direct observations $\boldsymbol{z}$. For the moment, let us assume the following true posterior predictive distribution for the task policy

$$\pi(\boldsymbol{x}) \sim \mathcal{N}\left(\mu(\boldsymbol{x}), \Sigma(\boldsymbol{x}, \boldsymbol{x}')\right) \tag{6.52}$$

In this case, we can compute an estimate of the task uncertainty as [84]

$$\sigma_r^2(\boldsymbol{x}_i) = \frac{1}{2}\left(\left(y_i - \mu(\boldsymbol{x}_i)\right)^2 + \sigma^2(\boldsymbol{x}_i)\right) \tag{6.53}$$

where $\sigma^2(\boldsymbol{x}_i) = \operatorname{diag}\left(\Sigma(\boldsymbol{x}_i, \boldsymbol{x}_i)\right)$. Thus, we can build a dataset for the task uncertainty $\mathcal{D}_z = (X, \boldsymbol{z})$ empirically from the demonstrations with

$$\boldsymbol{z} = \log\left(\frac{1}{2}\left(\left(\boldsymbol{y} - \mu(X)\right)^2 + \sigma^2(X)\right)\right) \tag{6.54}$$

Then, we can obtain the most likely log-uncertainty $(\widetilde{\boldsymbol{z}}, \widetilde{\boldsymbol{z}}_*)$ from equations (6.50) and (6.51). Finally, to recover the task uncertainty note that we cannot directly exponentiate $\widetilde{\boldsymbol{z}}$ and $\widetilde{\boldsymbol{z}}_*$ since

$$\mathbb{E}\left[\sigma_r^2(X)\right] = \mathbb{E}\left[\exp\left(\widetilde{\boldsymbol{z}}\right)\right] = \int \exp(\boldsymbol{z}) \cdot \mathcal{N}\left(\widetilde{\boldsymbol{z}}, \sigma_z^2(X)\right) d\boldsymbol{z} \tag{6.55}$$

By direct exponentiation we would under-predict the true uncertainty levels by introducing a bias from the transformation [85]. Integral (6.55) has an analytical solution, since the result is the mean of the Log Normal distribution. That is,

$$\mathbb{E}\left[\sigma_r^2(X)\right] = \exp\left(\widetilde{\boldsymbol{z}} + \frac{\sigma_z^2(X)}{2}\right) \tag{6.56}$$

However, remember that for obtaining the samples of the log-uncertainty $\boldsymbol{z}$ we have assumed the posterior predictive distribution (6.52), which we cannot compute without knowing the task uncertainty. For this reason, the optimization procedure has to be performed iteratively. Initially, we compute the posterior predictive distribution of $\pi(\boldsymbol{x})$ as a standard Gaussian Process. In this way, we can obtain a first estimate of the log-uncertainty and compute the heteroscedastic Gaussian Process. Then, we evaluate the similarity between the initial standard Gaussian Process and the heteroscedastic one. If they are very different, we take as our new estimate of $\pi(\boldsymbol{x})$ the heteroscedastic Gaussian Process and update the log-uncertainty, repeating the steps until convergence. The procedure is summarized in Algorithm 4.

Algorithm 4: Most Likely Heteroscedastic Gaussian Process

Input: $\mathcal{D} = (X, \boldsymbol{y})$ (demonstrations), X_* (prediction inputs),
$m_f(), k_f(,)$ ($f(\boldsymbol{x})$ prior mean and covariance),
$m_z(), k_z(,)$ ($z(\boldsymbol{x})$ prior mean and covariance),
$c(,)$ (uncertainty correlation)

`/* Train a standard Gaussian Process on` $\mathcal{D}$ `*/`

$\boldsymbol{\mu}, \boldsymbol{\Sigma} = \mathrm{GP}(\mathcal{D}, k_f, m_f)$;

while *True* **do**

`/* Estimate empirically the latent uncertainty */`

$\boldsymbol{z} = \log\left[\frac{1}{2}\left((\boldsymbol{y} - \boldsymbol{\mu})^2 + \mathrm{diag}\left(\boldsymbol{\Sigma}\right)\right)\right] \longrightarrow \mathcal{D}_z = (X, \boldsymbol{z})$;

`/* Train a standard Gaussian Process on` $\mathcal{D}_z$ `*/`

$\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z = \mathrm{GP}(\mathcal{D}_z, k_z, m_z)$;

`/* Train a heteroscedastic Gaussian Process on` $\mathcal{D}$ `*/`

$\sigma_r^2(X) = \exp\left(\boldsymbol{\mu}_z + \mathrm{diag}\left(\boldsymbol{\Sigma}_z\right)/2\right) \longrightarrow k_r(X, X) = \sigma_r^2(X) c(X, X)$;

$\boldsymbol{\mu}', \boldsymbol{\Sigma}' = \mathrm{HeteroscedasticGP}(\mathcal{D}, m_f, k_f, k_r)$;

`/* Evaluate convergence */`

if $\|\boldsymbol{\mu} - \boldsymbol{\mu}'\| < \varepsilon_\mu$ ***and*** $\|\boldsymbol{\Sigma} - \boldsymbol{\Sigma}'\| < \varepsilon_\Sigma$ **then**

break;

else

$\boldsymbol{\mu}, \boldsymbol{\Sigma} = \boldsymbol{\mu}', \boldsymbol{\Sigma}'$;

end

end

$\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_* = \mathrm{HeteroscedasticGP}(\mathcal{D}, X_*, m_f, k_f, k_r)$;

return: $\boldsymbol{\mu}_*$ (posterior mean), $\boldsymbol{\Sigma}_*$ (posterior covariance)

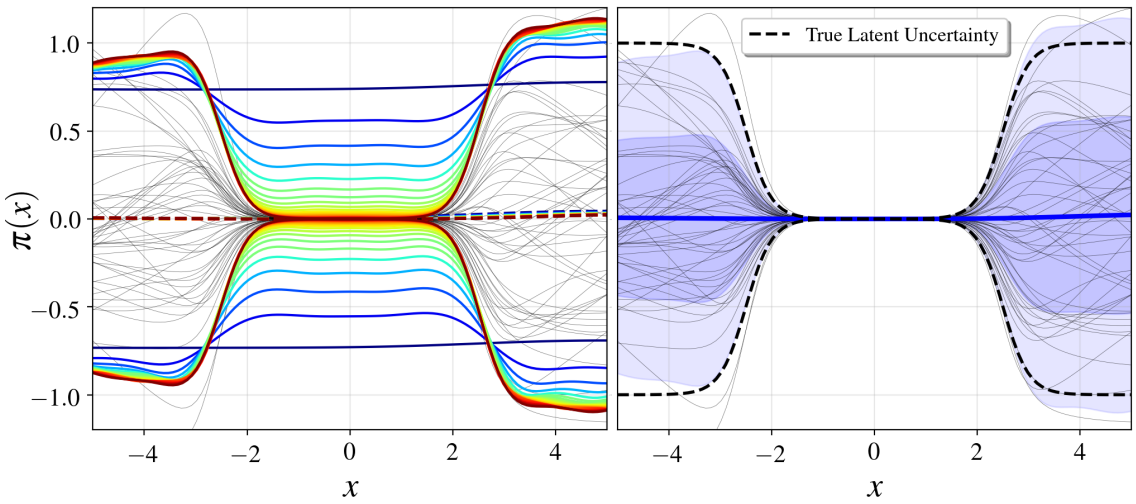

Figure 6.10: On the left, an illustration of the iterative optimization procedure summarized in Algorithm 4. The solid lines represent the evolution of two times the estimate of the standard deviation and the dashed lines the evolution of mean. Blue colors correspond to the first iterations while red colors correspond to the last ones. On the right, the resulting most likely heteroscedastic GP along with the true latent uncertainty.

In Figure 6.10, we illustrate the computation of the most likely heteroscedastic Gaussian Process using Algorithm 4. We can see that initially, the estimate of the task uncertainty is constant along with the task and it iteratively evolves towards the true value. Note that the estimation of the mean does not vary significantly, since the initial standard Gaussian Process already provides a good approximation.

6.2.2 Temporal Alignment of Demonstrations

The learned policy can be either time-dependent or time-invariant. As the name would suggest, time-dependent policies rely on time. These are potentially more expressive than their time-invariant counterparts and are capable of capturing strategies that vary with time and involve time-based requirements. For instance, time-dependent policies provide a straightforward mechanism to ensure that the reproduced behavior aligns with the demonstration in terms of speed and duration.

However, for robot manipulation tasks we are sometimes interested in capturing general strategies that are independent of time $t \in \mathbb{R}^+$. That is, in learning a time-invariant policy. These have only an implicit dependence on time just to capture sequential aspects of behavior using a phase variable $\tau \in \mathbb{R}$. The problem is that in general, it is very difficult for a human to repeat the demonstrations with the same velocity and accelerations. As a result, there are distortions and shifts in time between trajectories that might lead to poor performance when inferring the task uncertainty with Gaussian Processes.

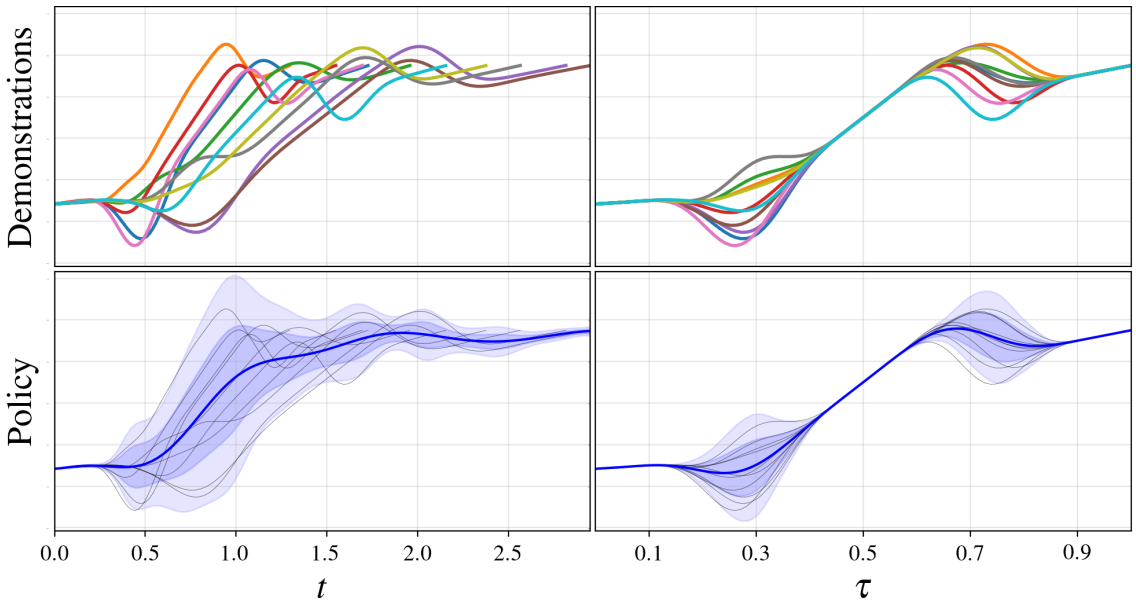

Figure 6.11: Importance of the temporal alignment of demonstrations for capturing the task uncertainty with time-invariant policies. This can be seen by comparing the left column, where demonstrations are affected by time distortions, against the right column, where demonstrations are aligned.

Figure 6.11 illustrates the question that we are addressing in this section. We can see that, by temporally aligning the demonstrations, we are able to capture the time-independent task uncertainty much more accurately. Correcting the time shifts in a preprocessing step improves significantly the policy inference process. First, we introduce the Dynamic Time Warping algorithm for temporal alignment of time series. Then, we present the task completion index, which allows aligning the demonstrations of robot manipulation tasks using this algorithm.

Dynamic Time Warping

The Dynamic Time Warping (DTW) algorithm [86] is a well-known technique to find an optimal alignment between two given (time-dependent) sequences under certain restrictions. Let $\boldsymbol{y}_1 = \{y_{11}, \ldots, y_{1n}\}$ and $\boldsymbol{y}_2 = \{y_{21}, \ldots, y_{2m}\}$ be two independent sequences of length n and m, respectively. To compare the similarity between two points of each sequence $y_{1i}, y_{2j} \in \mathcal{Y}$, one needs a local cost measure, sometimes also referred to as local distance measure, which is defined to be a function

$$d : \mathcal{Y} \times \mathcal{Y} \longrightarrow \mathbb{R}^+ \tag{6.57}$$

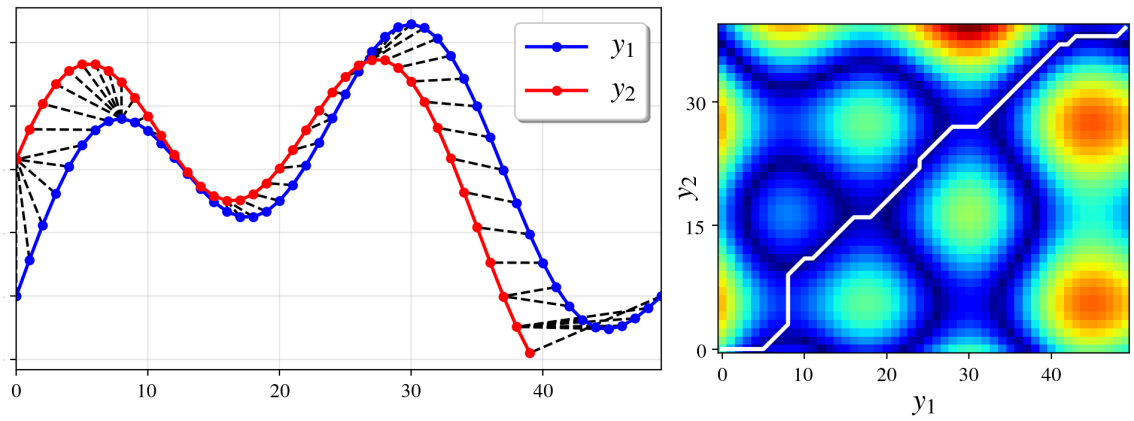

Figure 6.12: On the left, matching (dashed black lines) between two temporal sequences $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, obtained with the DTW algorithm. On the right, the corresponding distance matrix D, using the Euclidean distance. Regions of low distance are indicated by blue colors and regions of high distance are indicated by red colors. The warping path is depicted as a white solid line.

Typically, $d(y_{1i}, y_{2j})$ is small (low distance) if y_{1i} and y_{2j} are similar to each other, and otherwise $d(y_{1i}, y_{2j})$ is large (high distance). Evaluating the local distance measure for each pair of elements of the sequences $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, one obtains the distance matrix $D \in \mathbb{R}^{n \times m}$, defined by $D(i, j) = d(y_{1i}, y_{2j})$. Then the goal is to find the alignment between $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ which minimal overall distance. Intuitively, such an optimal alignment can be seen as a path that runs along a "valley" of low cost within the cost matrix D. See Figure 6.11 for an illustration.

We can formalize the notion of alignment with the following definition. An (n, m)-warping path (or simply warping path) is a sequence $p = \{p_1, \ldots, p_l\}$ of pairs $p_k = (i_k, j_k)$ defining the alignment between $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ that satisfies the following three conditions:

1. **Boundary Condition**: $p_1 = (1, 1)$ and $p_l = (n, m)$
2. **Monocity Condition**: $i_1 \leq i_2 \leq \cdots \leq i_l$ and $j_1 \leq j_2 \leq \cdots \leq j_l$
3. **Continuity Condition**: $p_{k+1} - p_k \in \{(1, 0), (0, 1), (1, 1)\}$

The boundary condition enforces that the first elements of $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ as well as the last elements of $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ are aligned to each other. In other words, the alignment refers to the entire sequences $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$. The monotonicity condition reflects the requirement of faithful timing: if an element in $\boldsymbol{y}_1$ precedes a second one this should also hold for the corresponding elements in $\boldsymbol{y}_2$, and vice versa. Finally, the continuity condition implies that no element in $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ can be omitted and there are no replications in the alignment. Figure 6.13 illustrates the three conditions.

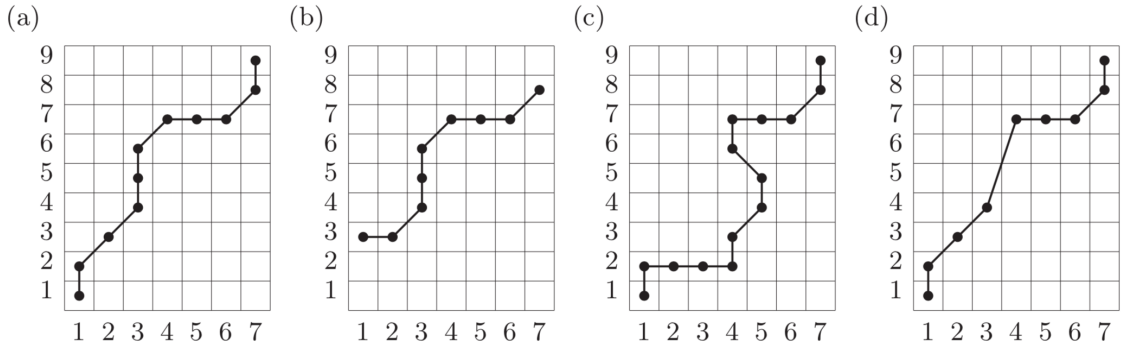

Figure 6.13: Illustration of the warping path conditions. (a) Admissible warping path. (b) The boundary condition is violated. (c) The monocity condition is violated. (d) The continuity condition is violated (after [86]).

The total distance $d_p(\boldsymbol{y}_1, \boldsymbol{y}_2)$ of a warping path p between $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ with respect to the local distance measure d is defined as

$$d_p(\boldsymbol{y}_1, \boldsymbol{y}_2) = \sum_{k=1}^{l} d(y_{1i_k}, y_{2j_k}) \tag{6.58}$$

An optimal warping path between $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ is a warping path p^* having minimal total distance among all possible warping paths. The DTW distance $\mathrm{DTW}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ between $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ is then defined as the total cost of p^*

$$\mathrm{DTW}(\boldsymbol{y}_1, \boldsymbol{y}_2) = d_{p^*}(\boldsymbol{y}_1, \boldsymbol{y}_2) \tag{6.59}$$

The usual approach to compute the optimal path p^* is based on a Dynamic Programming algorithm. We define the prefix sequences $\boldsymbol{y}_1(1:i) = \{y_{11}, \ldots, y_{1i}\}$ and and $\boldsymbol{y}_2(1:j) = \{y_{2j}, \ldots, y_{2j}\}$ and set

$$\gamma(i,j) = \mathrm{DTW}\left(\boldsymbol{y}_1(1:i), \boldsymbol{y}_2(1:j)\right) \tag{6.60}$$

All the entries $\gamma(i,j)$ define a $n \times m$ matrix, which is also referred to as the cumulative distance matrix. Each element represents the minimum total distance to be traveled by the path to reach (i,j). This matrix can be computed efficiently in a recursive fashion thanks to the following identities

- $\gamma(i,1) = \sum_{k=1}^{i} d(y_{1i}, y_{21})$ for $i = 1, \ldots, n$
- $\gamma(1,j) = \sum_{k=1}^{j} d(y_{11}, y_{2k})$ for $j = 1, \ldots, m$
- $\gamma(i,j) = d(y_{1i}, y_{2j}) + \min\left\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\right\}$

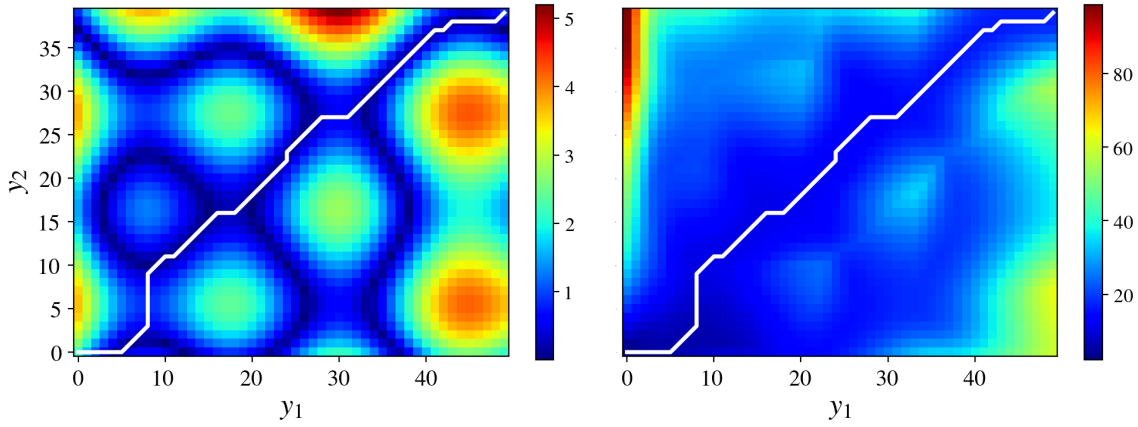

Figure 6.14: On the left, cost matrix D as in Figure 6.12. On the right, the corresponding accumulated cost matrix γ along with the optimal warping path p^* (white line).

Once we have the cumulative distance matrix, we can compute the optimal warping path following the procedure summarized in Algorithm 5. An example is shown in Figure 6.14, where the optimal warping path p^* along with the cumulative distance matrix γ for the sequences in Figure 6.12 are depicted.

Algorithm 5: Dynamic Time Warping

Input: γ (cumulative distance matrix)

$p^* = \{\}$;
$i = \text{rows}(\gamma),\ j = \text{columns}(\gamma)$;
while $i > 1$ *and* $j > 1$ **do**
  if $i = 1$ **then**
    $j = j - 1$;
  else if $j = 1$ **then**
    $i = i - 1$;
  else
    $q = \min\{\gamma(i-1, j), \gamma(i, j-1), \gamma(i-1, j-1)\}$;
    if $q = \gamma(i-1, j-1)$ **then**
      $i = i - 1, \quad j = j - 1$;
    else if $q = \gamma(i-1, j)$ **then**
      $i = i - 1$;
    else
      $j = j - 1$;
    end
  end
  $p^* = \{p^*, (i, j)\}$;
end
return: p^* (optimal warping path)

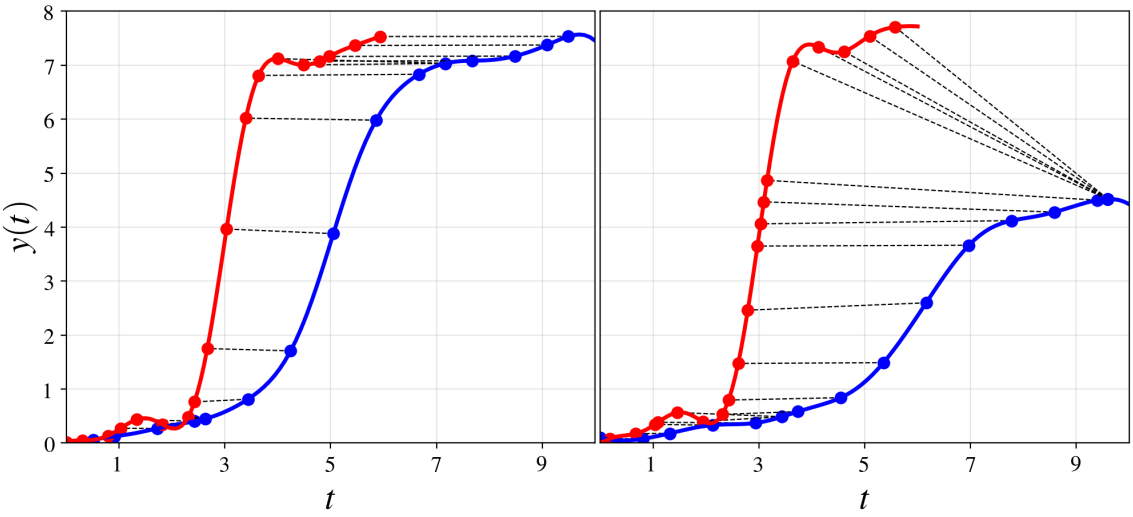

Figure 6.15: Warped demonstrations using the DTW algorithm and the Euclidean distance (6.61). On the left column, the demonstrations follow almost an identical path, being matched successfully. On the right column, they follow different paths, and the algorithm fails to warp them correctly.

Task Completion Index

The distance function d required by the Dynamic Time Warping algorithm can vary with the application. In robot learning from demonstration, the most common is the Euclidean distance between the demonstrated trajectories

$$d(y_{1i}, y_{2j}) = \|y_{1i} - y_{2j}\| \tag{6.61}$$

However, this relies on the assumption that the manipulation task must be performed always following the same path. An example is shown in Figure 6.15. The trajectories can be interpreted as demonstrations of a pick-and-place task where the object is placed at a different height. Using the Euclidean distance as a similarity measure between each point of the sequence, intermediate points for placing the object at a higher level are mapped to ending points of the demonstration at a lower level, since they are the closest in terms of (6.61). This leads to an erroneous warping, as we can see in the example.

We need to define a different similarity measure, suitable for warping correctly the demonstrations of a manipulation task where different paths are possible. We propose to perform the warping in terms of the task completion index ζ

$$d(y_{1i}, y_{2j}) = \|\zeta_{1i} - \zeta_{2j}\| \tag{6.62}$$

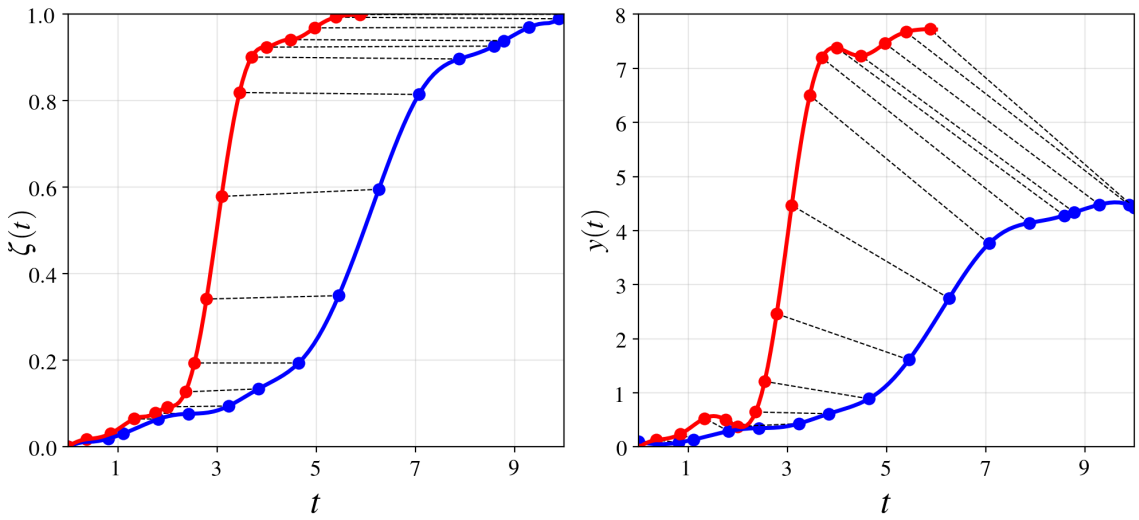

Figure 6.16: Warped demonstrations (on the right) using the DTW algorithm in the task completion index (6.63) space (on the left).

The task completion index (TCI) can be interpreted as a measure of the portion of the trajectory that has been covered for task completion. We define it as

$$\zeta(t) = \frac{\int_0^t \|dy(\tau)/d\tau\| d\tau}{\int_0^T \|dy(\tau)/d\tau\| d\tau} \tag{6.63}$$

where $y(\tau)$ refers to the value of the demonstrated trajectory $\boldsymbol{y}$, of duration T, at time instant τ. The numerator encodes the distance traveled until time instant t, while the denominator is equivalent to the total distance to cover until the completion of the task. Note that the TCI fulfills the following

$$0 = \zeta(0) \leq \zeta(t) \leq \zeta(T) = 1 \tag{6.64}$$

Alternatively, it can also be defined in discrete form as

$$\zeta_i = \frac{\sum_{k=1}^{i} |y_{k+1} - y_k|}{\sum_{k=1}^{n-1} |y_{k+1} - y_k|} \tag{6.65}$$

In Figure 6.16, we show that by performing the warping in the TCI space, we are able to warp the demonstrations in Figure 6.15 correctly.

6.3 Task Variables

An important aspect of learning from demonstrations is to enable robots to acquire skills that can be adapted to new situations. Such generalization capability can be achieved by associating the task variables, that describe the context under which the demonstrations are performed; with the movement variables, that describe the skill [35]. Encoding the policy with Gaussian Processes, we can consider the task variables as the inputs and the movement variables as the outputs. Then, given a set of demonstrations, the model can learn the constraints and requirements of the manipulation task from a general perspective. Being capable of retrieving an adaptive motion for a previously unseen context, described by a new set of task variables. These can represent a wide range of context features, for instance, the state of the environment, the robot configuration, positions of objects, etc. They are collected by the robot and can either be fixed or vary while the motion is executed.

The task variables can be either continuous or discrete. However, the standard Gaussian Process formulation assumes that the input variables are continuous. As a first approach to overcome this issue, one might think that we could train a distinct Gaussian Process model for each possible combination of the discrete variables. Then, while executing the task, the robot could select which model fits the current context. However, this method ignores possible correlations and becomes infeasible as the dimensionality of the discrete variables set grows, since the number of possible models increases exponentially.

The key ingredient for Gaussian Process design is the covariance function. Defining the kernel adequately is essential for capturing the topology of the input space, as we have seen for instance, in Section 6.1.3. In this section we show how to include discrete task variables, which can be either integer (Section 6.3.1) or categorical (Section 6.3.2) in the learned policy, focusing on the design of the covariance structure. Finally, in Section 6.3.3, we discuss how to incorporate discrete and continuous variables simultaneously in the Gaussian Process model.

6.3.1 Integer Task Variables

An integer variable $\iota \in \mathbb{Z}$ is a discrete variable with ordered levels. The values of the integer variable can be viewed as a discretization of a continuous one $\rho \in \mathbb{R}$. The key to designing suitable covariance functions is realizing that for integers, unlike categorical variables, the notion of order is preserved. Therefore, the concept of similarity between input locations in terms of the distance between points exploited in kernels for continuous variables can also be applied for discrete integer variables. Then, we can easily adapt the stationary kernels defined for real inputs to integers.

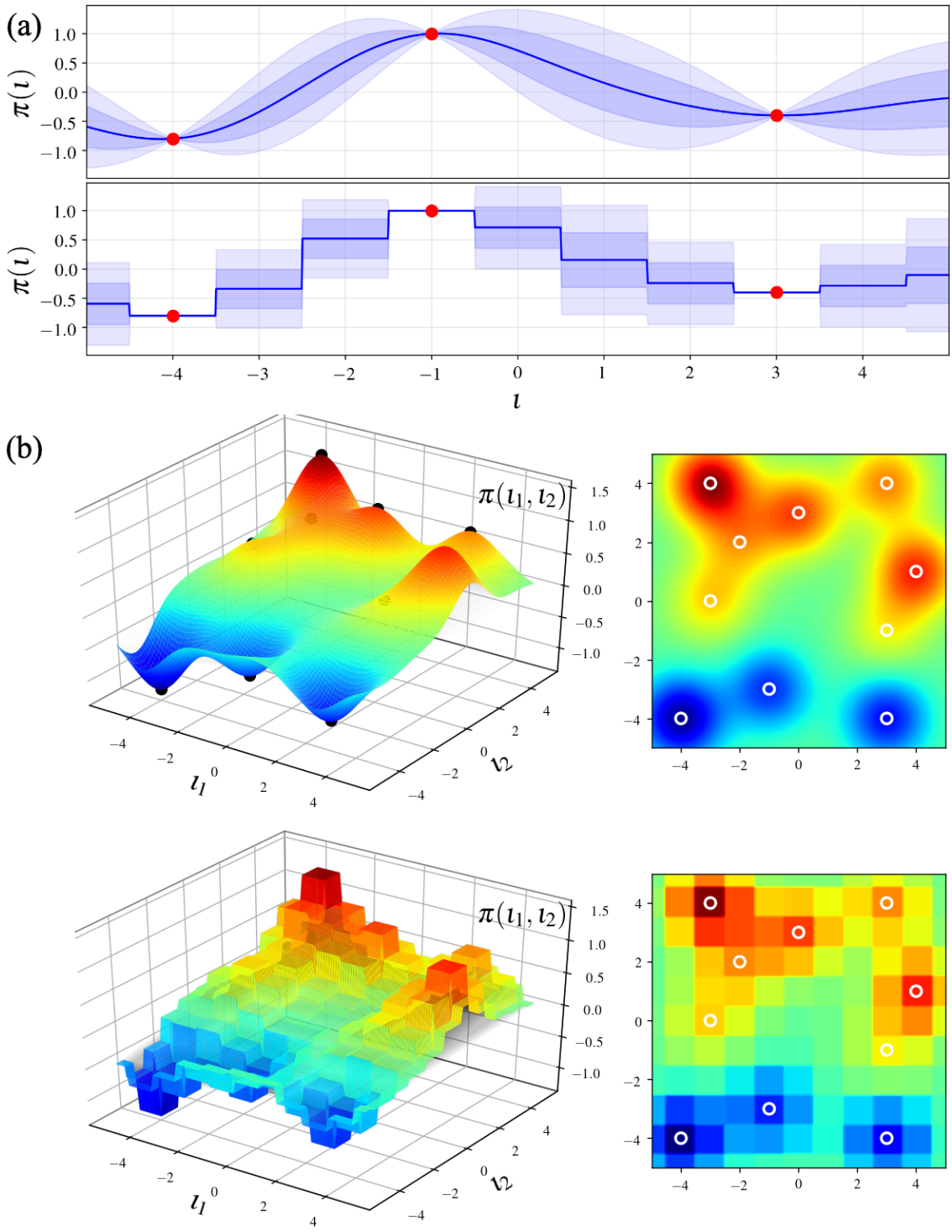

Figure 6.17: Comparison between Gaussian Processes over real input variables $\iota \in \mathbb{R}$ and discrete integers $\iota \in \mathbb{Z}$. (a) One-dimensional input. (b) Two-dimensional input. The training points are depicted as dots. Note that for obtaining an interpretable graphical representation we have depicted the discrete policy defined over the real plane. However, only those values at integer input locations are meaningful.

We can define a non-decreasing transformation

$$F : \mathbb{Z} \longrightarrow \mathbb{R} \tag{6.66}$$

such that the order is preserved, which projects the discrete variable into a continuous space. Therefore, a kernel $k_{\mathbb{Z}}$ for integer variables can be obtained from a kernel for real variables $k_{\mathbb{R}}$ by

$$k_{\mathbb{Z}}(\iota, \iota') = k_{\mathbb{R}}(F(\iota), F(\iota')) \tag{6.67}$$

In the general case, F is piecewise-linear. However, common warping functions are based on the cumulative distribution of a uniform, normal or lognormal random variable $F : \mathbb{Z} \longrightarrow [0, 1]$. Note that when $k_{\mathbb{R}}$ depends on the distance $\|x - x'\|$, then $k_{\mathbb{Z}}$ depends on the distance between ι, ι' distorted by F.

In order to allow negative correlations one may choose, for instance, the cosine correlation kernel on $[0, \beta)$, being $\beta \in (0, \pi]$ a fixed parameter tuning the minimal correlation value

$$k_{\mathbb{Z}}(\iota, \iota') = \cos(F(\iota) - F(\iota')) \tag{6.68}$$

An example of a Gaussian Process over discrete variables, compared with its continuous counterpart is depicted in Figure 6.17. We can see that the policy over integers is analogous to a discretization of the policy over reals, taking only values at the discrete input locations.

6.3.2 Categorical Task Variables

A categorical variable $\kappa \in \mathbb{K}$ is a discrete variable where there is not a notion of order between levels, in contrast with integer variables. Thus, we cannot use kernels based on the distance between input variables as we usually do for Gaussian Processes. What we have in a categorical input space $\mathbb{K}$ is the notion of equality $=$ and inequality $\neq$. The question then comes down to constructing a valid kernel based on these operations.

The usual approach for dealing with this type of variable in machine learning is one-hot encoding. Basically, this consists of adding as many extra input variables as different values the categorical variable can take. Then, each category is encoded by a '1' on the corresponding input and a '0' in all the remaining ones. Although it might be appealing for its simplicity, the major drawback is that the dimensionality of the input can increase dramatically.

The most parsimonious kernel parametrization is the compound symmetry (CS) covariance structure

$$k_{\mathbb{K}}(\kappa, \kappa') = \begin{cases} v & \text{if } \kappa = \kappa' \\ c & \text{if } \kappa \neq \kappa' \end{cases} \qquad \text{with} \;\; v \geq 0 \;\; \text{and} \;\; v \geq c \tag{6.69}$$

where v is the variance and c is the covariance. All pairs of categories are treated equally, being the similarity maximum for two equal input points, and minimum for different ones. This is an important limitation when the categorical variable can take a large number of values. A more flexible parametrization can be obtained by considering groups of categories [87]. Let the discrete categorical set be partitioned into G groups, and $g(\kappa)$ be the group number corresponding to the value κ of the categorical variable. The covariance function can be expressed as

$$k_{\mathbb{K}}(\kappa, \kappa') = \begin{cases} v & \text{if } \kappa = \kappa' \\ c_{g(\kappa), g(\kappa')} & \text{if } \kappa \neq \kappa' \end{cases} \tag{6.70}$$

where for all $i, j \in \{1, \ldots, G\}$, the terms $c_{i,i}/v$ are within-group correlations, and $c_{i,j}$ $(i \neq j)$ are between-group correlations. Note that additional constraints on v and $c_{i,j}$ are required to ensure that $k_{\mathbb{K}}$ is a valid kernel function. The corresponding Gram matrix $\boldsymbol{K}$, written in block form is

$$\boldsymbol{K}(\boldsymbol{\kappa}, \boldsymbol{\kappa}') = \begin{pmatrix} \boldsymbol{W}_1 & \ldots & \boldsymbol{B}_{1,G} \\ \vdots & \ddots & \vdots \\ \boldsymbol{B}_{G,1} & \ldots & \boldsymbol{W}_G \end{pmatrix} \tag{6.71}$$

where diagonal blocks $\boldsymbol{W}_g$ and constant off-diagonal blocks $\boldsymbol{B}_{g,g'}$ encode within-group and between-group covariances respectively. The necessary and sufficient conditions [87] for $\boldsymbol{K}$ to be positive semidefinite are:

- $\boldsymbol{W}_g$ is positive semidefinite $\forall\; g = 1, \ldots, G$
- $\boldsymbol{W}_g - \overline{W}_g \boldsymbol{J}_g$ is positive semidefinite $\forall\; g = 1, \ldots, G$

where $\boldsymbol{J}_g$ is a matrix of ones with the size of $\boldsymbol{W}_g$ and $\overline{W}_g$ the average of its elements. An example of a Gaussian Process over a binary categorical variable compared with its continuous counterpart is shown in Figure 6.18. We can see that since we only have two possible values of κ, the resulting policy is also binary.

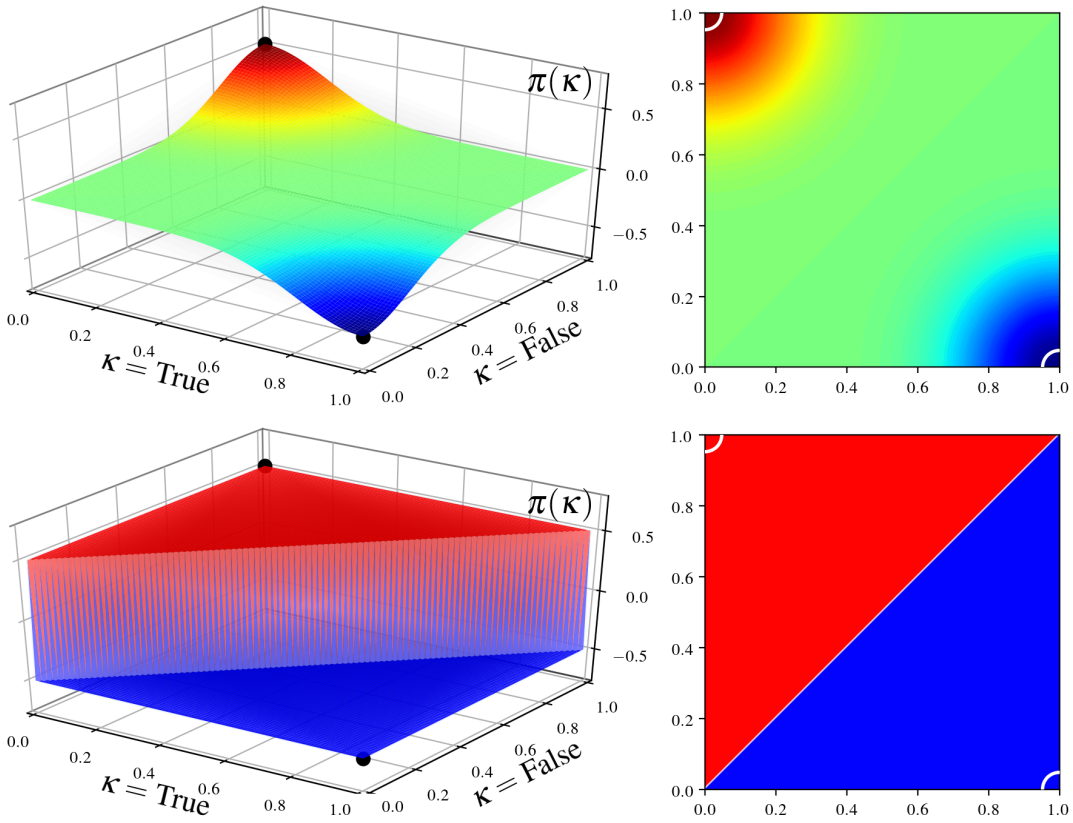

Figure 6.18: Comparison between Gaussian Processes over real input variables $\kappa \in \mathbb{R}$ and discrete categorical variables $\kappa \in \mathbb{K}$. The training points are depicted as black and white dots. Note that for obtaining an interpretable graphical representation we have encoded κ in the real plane as $(1, 0) = \text{True}$ and $(0, 1) = \text{False}$. However, only the policy at these input locations is meaningful.

6.3.3 Combining Real, Integer, and Categorical Variables

So far we have seen how to include discrete integer, categorical and real variables separately in a Gaussian Process. Now we address how to combine them in a single model. Without loss of generality, we can consider the three-dimensional input

$$\boldsymbol{x} = (\rho,\, \iota,\, \kappa) \in \mathcal{X} = \mathbb{R} \times \mathbb{Z} \times \mathbb{K} \tag{6.72}$$

Focusing the modeling effort on the covariance structure, kernels for inputs with multiple dimensions on $\mathcal{X}$ can be obtained by combining one-dimensional kernels [88] on $\mathbb{R}$, $\mathbb{Z}$ and $\mathbb{K}$. Standard valid combinations are the (1) product, (2) sum or (3) ANOVA across the different input dimensions.

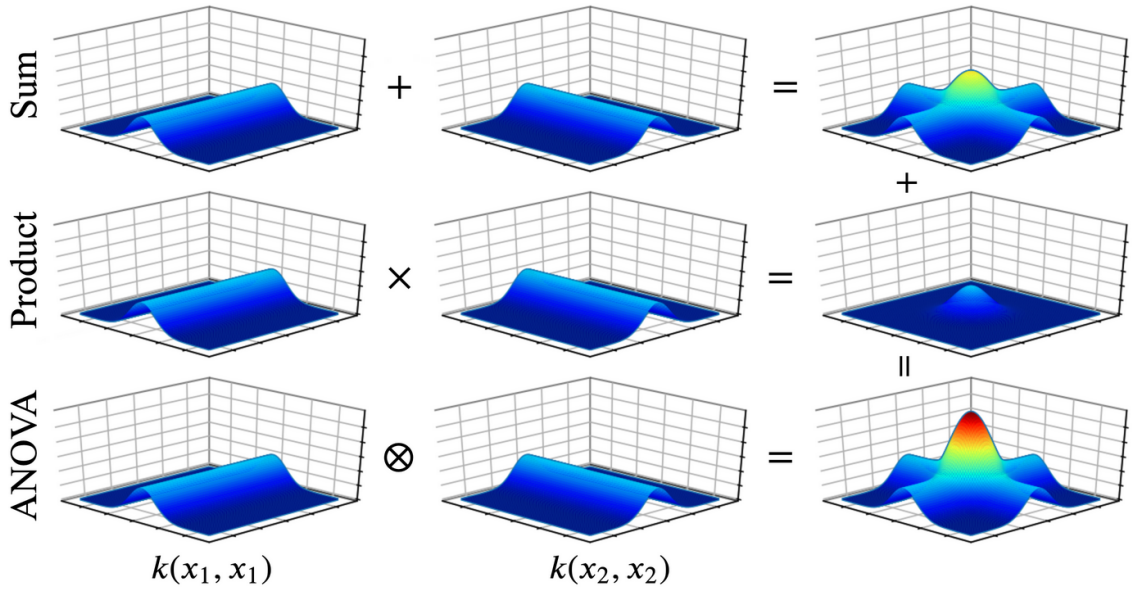

Figure 6.19: Sum (6.73), product (6.74) and ANOVA (6.75) composition across two input dimensions x_1 and x_2. The latter is equivalent to the sum of the first two.

The sum composition can be written as

$$k(\boldsymbol{x}, \boldsymbol{x}') = k_{\mathbb{R}}(\rho, \rho') + k_{\mathbb{Z}}(\iota, \iota') + k_{\mathbb{K}}(\kappa, \kappa') \tag{6.73}$$

Roughly speaking, adding two kernels can be thought of as an OR operation. That is, the resulting kernel will have a high value if any of the base kernels have a high value. On the other hand, the product composition is expressed as

$$k(\boldsymbol{x}, \boldsymbol{x}') = k_{\mathbb{R}}(\rho, \rho') \cdot k_{\mathbb{Z}}(\iota, \iota') \cdot k_{\mathbb{K}}(\kappa, \kappa') \tag{6.74}$$

This is the standard way of combining kernels across multiple dimensions. Roughly speaking, it can be thought of as an AND operation. That is, the resulting kernel will have a high value only if all the base kernels have a high value. Finally, the ANOVA composition is written as

$$k(\boldsymbol{x}, \boldsymbol{x}') = [1 + k_{\mathbb{R}}(\rho, \rho')] \cdot [1 + k_{\mathbb{Z}}(\iota, \iota')] \cdot [1 + k_{\mathbb{K}}(\kappa, \kappa')] - 1 \tag{6.75}$$

It can be seen as a generalization of (6.73) and (6.74), since it captures all the interactions in terms of sums and products between the input variables. This allows more flexibility in the model. The operation is equivalent to the tensor product. An illustration of the presented kernel compositions for a two-dimensional input is shown in Figure 6.19. We can see that the ANOVA composition is equivalent to the combination of the sum and the product.

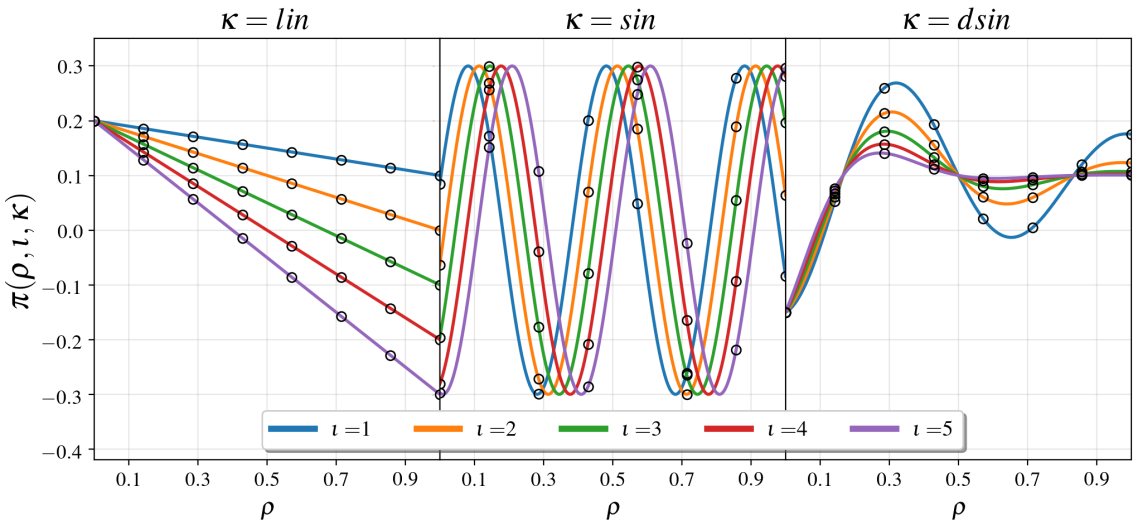

Figure 6.20: Task which depends on real ρ, integer ι and categorical κ variables encoded by function (6.76). The samples used as demonstration data are depicted as black dots.

In order to illustrate the combination of real, categorical, and integer variables in the same policy, consider a task represented deterministically by the following function

$$\pi(\rho, \iota, \kappa) = \begin{cases} 0.2 - \rho \cdot \iota/10 & \text{if } \kappa = \text{lin} \\ 0.3 \sin(5\pi \cdot \rho - \iota/2 + \pi/4) & \text{if } \kappa = \text{sin} \\ 0.1 + 0.3 \sin(3\pi \cdot \rho - \pi/2) \exp(1.2 \cdot \rho \cdot \iota) & \text{if } \kappa = \text{dsin} \end{cases} \tag{6.76}$$

with $\rho \in [0, 1]$, $\iota \in \{1, 2, 3, 4, 5\}$ and $\kappa \in \{\text{lin}, \text{sin}, \text{sin}\}$. We aim at reconstructing this task policy from demonstrations. As training data we take the samples depicted in Figure 6.20. We can see that the path varies greatly with κ, being either linear (lin), a sinusoid (sin) or a damped sinusoid (dsin). On the other hand, slight variations are explained by the integer variable ι.

For the kernel design, we consider the ANOVA composition (6.75), a squared-exponential kernel for the integer and real inputs, and a block compound symmetry kernel (6.70) for the categorical input. The resulting covariance matrix after optimizing the hyperparameters by maximum likelihood is shown in Figure 6.21a). We can clearly see the blocks of the categorical kernel, being almost uncorrelated $\kappa = \text{lin}$ and $\kappa = \text{sin}$, and quite correlated $\kappa = \text{sin}$ and $\kappa = \text{dsin}$. Also, note that each block is at the same time divided into five other blocks, corresponding to the different values of ι. The resulting mean of the policy, inferred from the demonstrations, is depicted in Figure 6.21b). We can observe that the Gaussian Process model is able to recover (6.76) using the proposed kernel design.

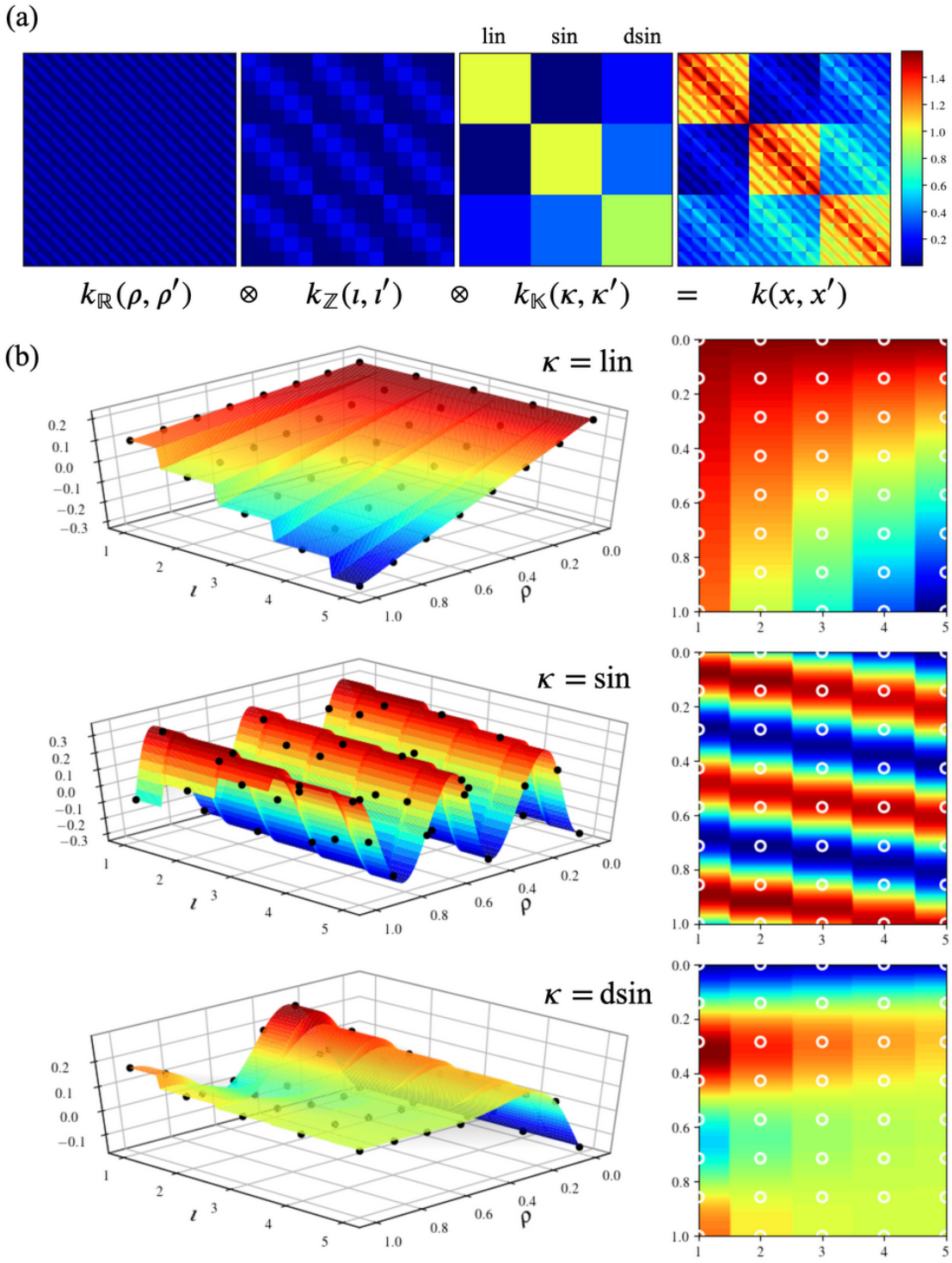

Figure 6.21: Policy inferred from the demonstrations of task (6.76). (a) Training covariance matrix after optimization of the hyperparameters by maximum likelihood, construced as the ANOVA composition (6.75) of the one-dimensional kernels $k_{\mathbb{R}}(\rho, \rho')$, $k_{\mathbb{Z}}(\iota, \iota')$ and $k_{\mathbb{K}}(\kappa, \kappa')$. (b) Posterior predictive mean of the policy after providing the training data depicted as black and white dots, using the covariance structure in (a).

6.4 Policy Adaptation

Human demonstrations can be used to retrieve a distribution of trajectories that the robot exploits to carry out a specific task. However, in dynamic and unstructured environments the robot also needs to adapt its motions when required. For example, if an obstacle suddenly occupies an area that intersects the robot's motion path, the robot is required to modulate its movement trajectory so that collisions are avoided. A similar modulation is necessary (e.g., in pick-and-place and reaching tasks) when the target varies its location during the task execution [49]. This can be achieved by introducing via-point constraints in the policy.

Besides the modulation of a single trajectory, another challenging problem arises when the robot is given a set of movement policies that only solve a fraction of the task. For instance, in a task where actions on different objects are required. The candidate policies can be exploited to compute a mixed trajectory, combining all of them into a single movement.

In this section, we do not focus on retrieving a policy of the task, but on its adaptation, once we have it. First, in Section 6.4.1, we discuss how to constraint it to pass through specified via-points. Then, in Section 6.4.2, we show how to combine multiple policies to retrieve a single representation that encompasses all of them.

6.4.1 Via-point Constraints

The modulation of the policy through via-point constraints is an important property to adapt the learned motion to new situations. Formally, let us define a set of M desired via-points

$$\mathcal{V} = \{\boldsymbol{x}_{vi}, y_{vi}\}_{i=1}^{M} = (X_v, \boldsymbol{y}_v) \tag{6.77}$$

Considering a stochastic representation of the policy, the problem is to find the posterior predictive distribution given the demonstrations dataset $\mathcal{D} = (X, \boldsymbol{y})$ and the via-point constraints $\mathcal{V}$ for the query input locations X_*. In probabilistic terms, this is equivalent to

$$\pi(X_*) \sim p\left(\pi(X_*) \mid \mathcal{D}, \mathcal{V}, X_*\right) \tag{6.78}$$

Assuming that the demonstrations are independent of the via-point constraints we can express (6.78) as

$$\pi(X_*) \sim p\left(\pi(X_*) \mid \mathcal{D}, X_*\right) \cdot p\left(\pi(X_*) \mid \mathcal{V}, X_*\right) \tag{6.79}$$

As we have seen throughout this chapter, the first term in (6.79) can be obtained using a Gaussian Process model. But, what about the second term? The idea behind imposing via-point constraints is that we want our policy $\pi(\boldsymbol{x})$ to retrieve action y_{vi} when the input is $\boldsymbol{x}_{vi}$. Probabilistically, we can write this as

$$\pi(\boldsymbol{x}_{vi}) \sim p(\pi(\boldsymbol{x}_{vi}) \mid \boldsymbol{x}_{vi}, y_{vi}) = \mathcal{N}\left(y_{vi}, \sigma_{vi}^2\right) \tag{6.80}$$

where the variance σ_{vi}^2 is selected based on the task requirements. For instance, if the robot needs to pass through y_i with high precision, we should assign a small variance. On the contrary, for via-points that allow for large tracking errors, we can set a high variance. Considering the whole set of via-points $\mathcal{V}$ instead of a single point, (6.80) is extended to a multivariate Gaussian distribution

$$\pi(X_v) \sim p\left(\pi(X_v) \mid \mathcal{V}\right) = \mathcal{N}\left(\boldsymbol{y}_v, \boldsymbol{\Sigma}_v\right) \tag{6.81}$$

What does this tell us about the conditional probability distribution at the query points $p\left(\pi(X_*) \mid \mathcal{V}, X_*\right)$? Note that this problem is analogous to a standard Gaussian Process regression, but considering an input-dependent variance σ_{vi}^2. In Section 6.2.1, we discussed how to embed this type of variance in the model. The difference, in this case, is that instead of inferring it from the demonstrations, we should specify it based on the task requirements. As we did in Section 6.2.1, assuming a null prior mean, the posterior distribution of the via-point constraints is given by

$$\pi(X_*) \sim p(\pi(X_*) \mid \mathcal{V}, X_*) = \mathcal{N}\left(\boldsymbol{\mu}_{v*}, \boldsymbol{\Sigma}_{v*}\right) \tag{6.82}$$

$$\boldsymbol{\mu}_{v*} = k(X_*, X_v)\left(k(X_v, X_v) + k_v(X_v, X_v)\right)^{-1} \boldsymbol{y}_v \tag{6.83}$$

$$\boldsymbol{\Sigma}_{v*} = k_v(X_*, X_*) - k(X_*, X_v)\left(k(X_v, X_v) + k_v(X_v, X_v)\right)^{-1} k(X_v, X_*) \tag{6.84}$$

encoding $k(,)$ the relation between the mean of the via-point constraints and the mean at an arbitrary input location, and $k_v(,)$ analogously for the variance. The latter can be expressed as

$$k_v(\boldsymbol{x}, \boldsymbol{x}') = \sigma_v(\boldsymbol{x})\sigma_v(\boldsymbol{x}')c(\boldsymbol{x}, \boldsymbol{x}') \tag{6.85}$$

where $\sigma_v(x_{vi}) = \sigma_{vi}$ specifies the strength of the via-point constraint and $c(\boldsymbol{x}, \boldsymbol{x}')$ defines the correlation between $\boldsymbol{x}$ and $\boldsymbol{x}'$. Then, if the posterior predictive distribution of the policy conditional on the demonstrations is given by

$$\pi(X_*) \sim p(\pi(X_*) \mid \mathcal{D}, X_*) = \mathcal{N}\left(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*\right) \tag{6.86}$$

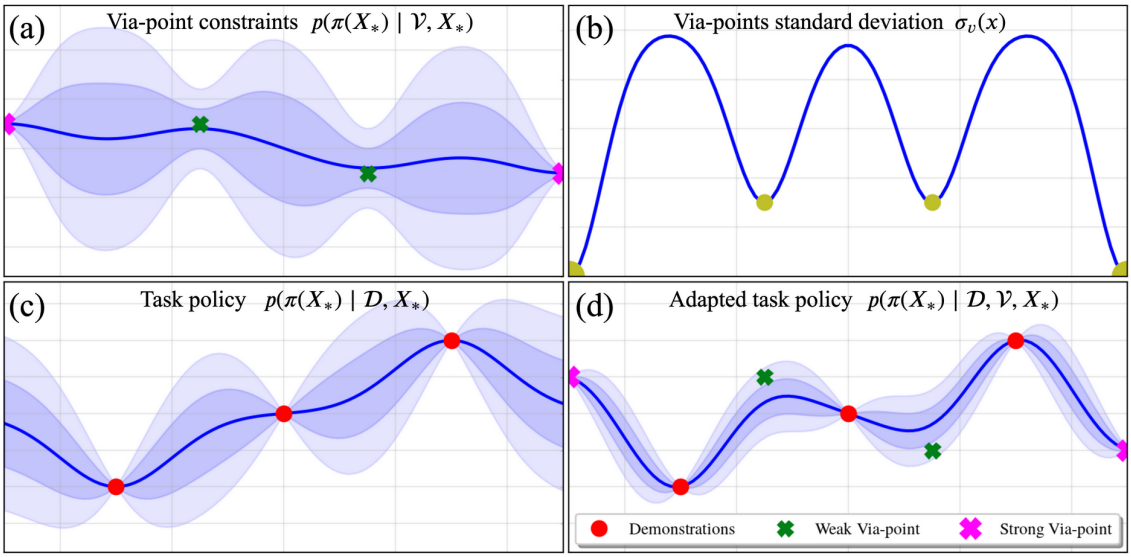

Figure 6.22: Illustrative example of the adaptation of a policy through via-points. In (a) we have the posterior distribution of the specified via-point constraints (6.82), considering the variance in (b). Then, in (d) we have the policy depicted in (c) after the modulation.

We have that (6.78) is then given by the product of two Gaussian distributions, which yields also a Gaussian distribution of the following form

$$\pi(X_*) \sim p\left(\pi(X_*) \mid \mathcal{D}, \mathcal{V}, X_*\right) = C^{-1} \cdot \mathcal{N}\left(\boldsymbol{\mu}'_*, \boldsymbol{\Sigma}'_*\right) \tag{6.87}$$

$$\boldsymbol{\mu}'_* = \left(\boldsymbol{\Sigma}_*^{-1} + \boldsymbol{\Sigma}_{v*}^{-1}\right)^{-1} \left(\boldsymbol{\Sigma}_*^{-1}\boldsymbol{\mu}_* + \boldsymbol{\Sigma}_{v*}^{-1}\boldsymbol{\mu}_{v*}\right) \tag{6.88}$$

$$\boldsymbol{\Sigma}'_* = \left(\boldsymbol{\Sigma}_*^{-1} + \boldsymbol{\Sigma}_{v*}^{-1}\right)^{-1} \tag{6.89}$$

$$C = \left|2\pi\left(\boldsymbol{\Sigma}_* + \boldsymbol{\Sigma}_{v*}\right)\right|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\left(\boldsymbol{\mu}_* - \boldsymbol{\mu}_{v*}\right)^T \left(\boldsymbol{\Sigma}_* + \boldsymbol{\Sigma}_{v*}\right)^{-1} \left(\boldsymbol{\mu}_* - \boldsymbol{\mu}_{v*}\right)\right) \tag{6.90}$$

where C is a normalization constant. The resulting distribution is a compromise between the via-point constraints and the demonstrated trajectories, weighted inversely by their variances. Thus, via-points with low variance modify significantly the learned policy and via-points with a high variance produce a more subtle modulation. Additionally, it is important to note that for modulating the policy we do not have to re-train the model, since we only need the posterior distribution at the query input locations.

An illustrative example of the adaptation of a policy through via-points is shown in Figure 6.22. Those via-points that have a smaller variance produce a stronger effect. Also, it should be remarked that a smooth specification of the variance profile $\sigma_v(\boldsymbol{x})$ is recommended for the via-point constraints. A sharp profile would modulate the policy only in a small region close to the via-points, appearing peaks in the mean trajectory, which is not desirable for a robot policy.

6.4.2 Combination of Policies

In some cases, decomposing a manipulation task in multiple sub-tasks allows for more flexibility in the learning process. We can train multiple simple policies and then combine them based on the current task requirements instead of learning a complex policy for each particular situation. Formally, given a set of M task policies

$$\mathcal{P} = \{\pi_i(\boldsymbol{x})\}_{i=1}^{M} \tag{6.91}$$

How can we encompass all of them in a single policy $\hat{\pi}(\boldsymbol{x})$? Intuitively, one can think that each policy encodes a series of task constraints. Thus, for obtaining a meaningful blend, the combined representation must encode all the task constraints simultaneously. Let each sub-policy $\pi_i()$ have associated the following posterior predictive distribution for the query inputs

$$\pi_i(X_*) \sim p(\pi_i(X_*) \mid \mathcal{D}_i, X_*) = \mathcal{N}\left(\boldsymbol{\mu}_{i*}, \boldsymbol{\Sigma}_{i*}\right) \tag{6.92}$$

where $\mathcal{D}_i$ is the corresponding demonstrations dataset. Assuming each set of task constraints $\pi_i()$ as independent, the posterior predictive distribution of the combined policy is given by

$$\hat{\pi}(X_*) \sim p(\hat{\pi}(X_*) \mid \mathcal{D}_1, \ldots, \mathcal{D}_M, X_*) = \prod_{i=1}^{M} p(\pi_i(X_*) \mid \mathcal{D}_i, X_*) \tag{6.93}$$

Since each sub-policy $\pi_i()$ is distributed according to a Gaussian distribution (6.92), the product (6.93) is again a Gaussian with the following mean and variance

$$\hat{\pi}(X_*) \sim \mathcal{N}\left(\hat{\boldsymbol{\mu}}_*, \hat{\boldsymbol{\Sigma}}_*\right) \tag{6.94}$$

$$\hat{\boldsymbol{\mu}}_* = \left(\sum_{i=1}^{M} \boldsymbol{\Sigma}_{i*}^{-1}\right)^{-1} \left(\sum_{i=1}^{M} \boldsymbol{\Sigma}_{i*}^{-1} \boldsymbol{\mu}_{i*}\right) \tag{6.95}$$

$$\hat{\boldsymbol{\Sigma}}_* = \left(\sum_{i=1}^{M} \boldsymbol{\Sigma}_{i*}^{-1}\right)^{-1} \tag{6.96}$$

We can see that the resulting combined policy is a compromise between the M policies, each weighted inverse to their variances. Thus, those task constraints with a lower uncertainty have a higher relative weight.

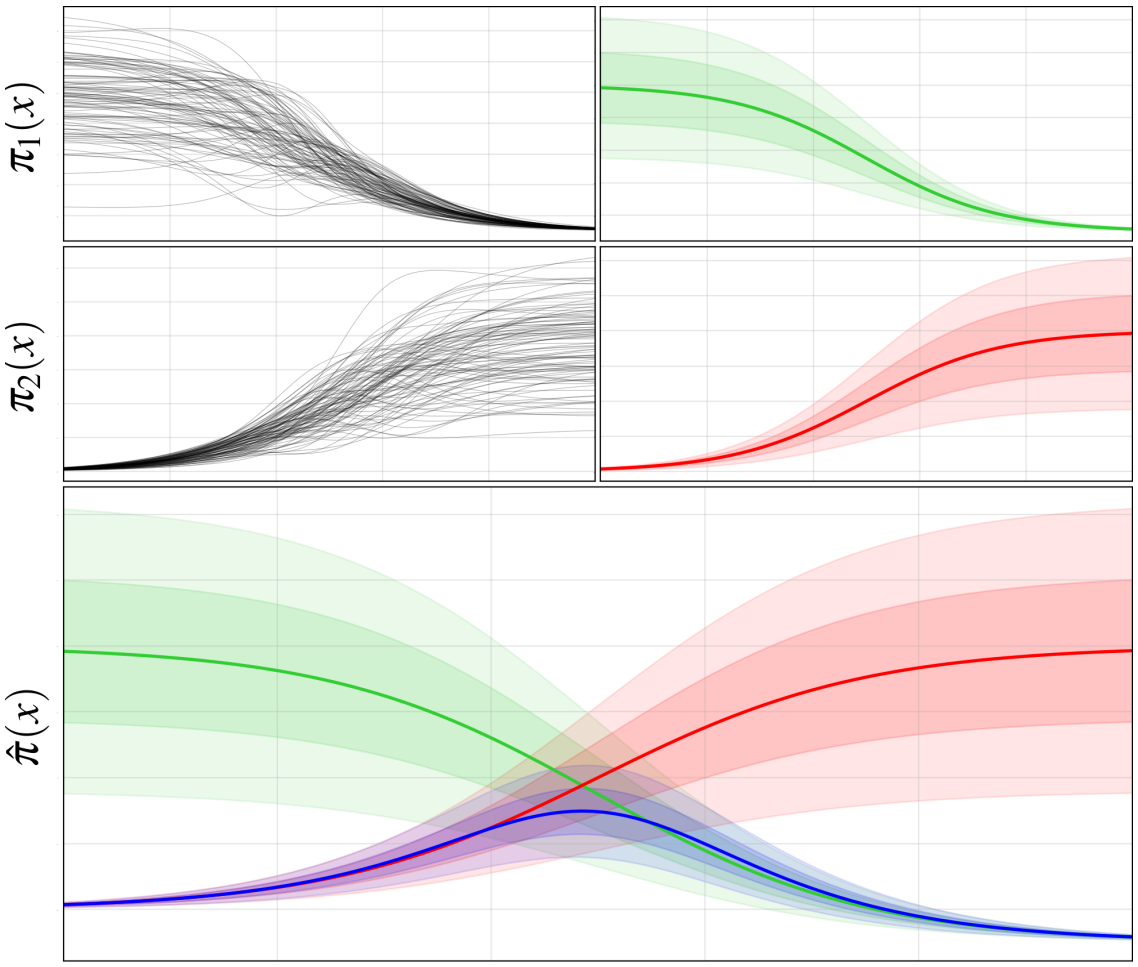

Figure 6.23: Combination of two task policies $\pi_1(x)$ and $\pi_2(x)$. In the first and the second rows, we have the independent extraction of two task constraints. In the third row, we have the combined policy $\hat{\pi}(x)$ (blue color).

Note that the introduction of via-point constraints, discussed in Section 6.4.1, can also be interpreted as the combination of two independent policies. The difference is that the policy associated with the via-point constraints is not given beforehand. Instead, we have to infer it from the specified via-points.

An illustrative example of the combination of two task policies is shown in Figure 6.23. Two demonstration datasets are used to train two independent models, $\pi_1(x)$ and $\pi_2(x)$. We can see that the former presents a high task uncertainty for small values of the input x, and a small uncertainty for high values. On the other hand, the opposite occurs for $\pi_2(x)$. We can see that the combined policy $\hat{\pi}(x)$ represents an intermediate behavior between the sub-policies, weighted inversely by their uncertainty. For low values of x is almost identical to $\pi_1(x)$ while for higher values it is almost identical to $\pi_2(x)$.

6.5 Inference and Prediction under Replication

When the demonstration dataset becomes large, the computational requirements of Gaussian Processes start to become important. Especially, when the number of training points N becomes larger than a thousand [6]. Using Gaussian Processes, mainly two operations are involved: prediction and inference. The former refers to the computation of the posterior predictive distribution

$$\boldsymbol{\pi}_* = \pi(X_*) \sim \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \tag{6.97}$$

$$\boldsymbol{\mu}_* = m(X_*) + k(X_*, X)k(X, X)^{-1}(\boldsymbol{y} - m(X)) = \boldsymbol{m}_* + \boldsymbol{K}_*^T \boldsymbol{K}^{-1}(\boldsymbol{y} - \boldsymbol{m}) \tag{6.98}$$

$$\boldsymbol{\Sigma}_* = k(X_*, X_*) + k(X_*, X)k(X, X)^{-1}k(X, X_*) = \boldsymbol{K}_{**} - \boldsymbol{K}_*^T \boldsymbol{K}^{-1} \boldsymbol{K}_* \tag{6.99}$$

On the other hand, inference refers to the evaluation of the log-likelihood for the optimization of the kernel hyperparameters $\boldsymbol{\theta}$

$$\mathcal{L} = \log p(\boldsymbol{y} \mid X, \boldsymbol{\theta}) = -\frac{1}{2}(\boldsymbol{y} - \boldsymbol{m})^T \boldsymbol{K}^{-1}(\boldsymbol{y} - \boldsymbol{m}) - \frac{1}{2}\log|\boldsymbol{K}| - \frac{N}{2}\log 2\pi \tag{6.100}$$

A question we could ask ourselves is which is the computational complexity of these calculations. This depends on the number of training N and prediction N_* points. To analyze it, we can apply the following rules

- Adding/substracting two $n \times m$ matrices takes $\mathcal{O}(nm)$ time.
- Multiplying and $n \times m$ matrix by a $m \times l$ matrix takes $\mathcal{O}(nml)$ time.
- Multiplying a $n \times m$ matrix by a vector of size m takes $\mathcal{O}(nm)$ time.
- Inverting a $n \times n$ matrix takes $\mathcal{O}(n^3)$ time.
- Computing the determinant of a $n \times n$ matrix takes $\mathcal{O}(n^3)$ time.

Using these results, we can see that regarding the prediction step, calculating the predictive mean $\boldsymbol{\mu}_*$ (6.98) takes $\mathcal{O}(N^3 + N_*N^2 + N_*N + N + N_*)$ time, and computing the predictive covariance $\boldsymbol{\Sigma}_*$ (6.99) takes $\mathcal{O}(N^3 + N_*N^2 + N_*^2N + N_*^2)$ time. With respect to the inference step, the computation of the log-likelihood takes $\mathcal{O}(2N^3 + N^2 + 3N)$ time. In practice we usually have that the number of training points is much larger than the number of query points $N \gg N_*$. Thus, the computational complexity of a Gaussian Process regression is approximately $\mathcal{O}(N^3)$, which scales dramatically with the amount of demonstration data.

For alleviating the computational complexity we can exploit the structure of replications i.e., repeated demonstrations for identical inputs [89]. These appear naturally in the context of learning from demonstration since replications are essential for capturing the task uncertainty. For the developement, we assume a heterocedastic Gaussian Process model, which we discuss in detail in Section 6.2.1. The standard prediction (6.98), (6.99) and inference (6.100) equations are changed to

$$\boldsymbol{\mu}_* = \boldsymbol{m}_* + \boldsymbol{K}_*^T \left(\boldsymbol{K} + \boldsymbol{R}\right)^{-1} \left(\boldsymbol{y} - \boldsymbol{m}\right) \tag{6.101}$$

$$\boldsymbol{\Sigma}_* = \boldsymbol{K}_{**} + \boldsymbol{R}_{**} - \boldsymbol{K}_*^T \left(\boldsymbol{K} + \boldsymbol{R}\right)^{-1} \boldsymbol{K}_* \tag{6.102}$$

$$\mathcal{L} = -\frac{1}{2} \left(\boldsymbol{y} - \boldsymbol{m}\right)^T \left(\boldsymbol{K} + \boldsymbol{R}\right)^{-1} \left(\boldsymbol{y} - \boldsymbol{m}\right) - \frac{1}{2} \log \left|\boldsymbol{K} + \boldsymbol{R}\right| - \frac{N}{2} \log 2\pi \tag{6.103}$$

where $\boldsymbol{R}$ and $\boldsymbol{R}_{**}$ are the covariance matrices that encode the task uncertainty at the training and query inputs, respectively, which for this section we assume diagonal. Now, let $\hat{\boldsymbol{x}}_i$ represent the i-th out of $n \ll N$ unique input locations in the demonstrations dataset, and $y_i^{(j)}$ be the j-th out of $a_i \geq 1$ replicates observed at $\hat{\boldsymbol{x}}_i$, where $\sum_{i=1}^{n} a_i = N$. Then, let

$$\hat{\boldsymbol{y}} = \begin{bmatrix} \hat{y}_1 & \dots & \hat{y}_n \end{bmatrix}^T, \qquad \hat{y}_i = a_i^{-1} \sum_{j=1}^{a_i} y_i^{(j)} \tag{6.104}$$

be the vector whose i-th element is the average of the observations at $\hat{\boldsymbol{x}}_i$. We now develop a map from the covariance matrices considering the N data points $\boldsymbol{K}_N$ and $\boldsymbol{R}_N$ to their unique-n counterparts $\boldsymbol{K}_n$ and $\boldsymbol{R}_n$. Without loss of generality, assume that the data are ordered so that

$$X = \begin{bmatrix} \hat{\boldsymbol{x}}_1 & \dots & \hat{\boldsymbol{x}}_1 & \dots & \hat{\boldsymbol{x}}_n \end{bmatrix} \tag{6.105}$$

where each input is repeated a_i times, and where $\boldsymbol{y}$ is stacked with observations on the a_i replicates in the same order. With X composed in this way, we have

$$\boldsymbol{K}_N = \boldsymbol{U}\boldsymbol{K}_n\boldsymbol{U}^T, \quad \boldsymbol{K}_{N*} = \boldsymbol{U}\boldsymbol{K}_{n*}, \quad \boldsymbol{U}^T\boldsymbol{R}_N\boldsymbol{U} = \boldsymbol{A}_n\boldsymbol{R}_n, \quad \boldsymbol{U}^T\boldsymbol{y} = \boldsymbol{A}_n\hat{\boldsymbol{y}} \tag{6.106}$$

where the matrices $\boldsymbol{U}$ and $\boldsymbol{A}_n$ fulfill the following identities

$$\boldsymbol{U} = \operatorname{diag}\left(\boldsymbol{1}_{a_1,1}, \dots, \boldsymbol{1}_{a_n,1}\right), \qquad \boldsymbol{A}_n = \operatorname{diag}\left(a_1, \dots, a_n\right), \qquad \boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{A}_n \tag{6.107}$$

denoting $\boldsymbol{1}_{k,l}$ a $k \times l$ matrix filled with ones.

We can write the prediction (6.101)(6.102) and inference (6.103) equations in terms of $\boldsymbol{K}_n$ and $\boldsymbol{R}_n$ making use of two well-known formulas, together comprising the Woodbury identity

$$(\boldsymbol{D}+\boldsymbol{U}\boldsymbol{B}\boldsymbol{V})^{-1} = \boldsymbol{D}^{-1} - \boldsymbol{D}^{-1}\boldsymbol{U}\left(\boldsymbol{B}^{-1}+\boldsymbol{V}\boldsymbol{D}^{-1}\boldsymbol{U}\right)^{-1}\boldsymbol{V}\boldsymbol{D}^{-1} \tag{6.108}$$

$$|\boldsymbol{D}+\boldsymbol{U}\boldsymbol{B}\boldsymbol{V}| = \left|\boldsymbol{B}^{-1}+\boldsymbol{V}\boldsymbol{D}^{-1}\boldsymbol{U}\right| \cdot |\boldsymbol{B}| \cdot |\boldsymbol{D}| \tag{6.109}$$

where $\boldsymbol{D}$ and $\boldsymbol{B}$ are invertible matrices. Using decomposition (6.106), and applying the Woodbury identity (6.101), taking $\boldsymbol{D} = \boldsymbol{R}_N$, $\boldsymbol{B} = \boldsymbol{K}_n$ and $\boldsymbol{V} = \boldsymbol{U}^T$, the predictive equations yield (see [89] for the full proof)

$$\boldsymbol{\mu}_* = \boldsymbol{m}_* + \boldsymbol{K}_{n*}^T\left(\boldsymbol{K}_n + \boldsymbol{A}_n^{-1}\boldsymbol{R}_n\right)^{-1}(\hat{\boldsymbol{y}} - \hat{\boldsymbol{m}}) \tag{6.110}$$

$$\boldsymbol{\Sigma}_* = \boldsymbol{K}_{**} + \boldsymbol{R}_{**} - \boldsymbol{K}_{n*}^T\left(\boldsymbol{K}_n + \boldsymbol{A}_n^{-1}\boldsymbol{R}_n\right)^{-1}\boldsymbol{K}_{n*} \tag{6.111}$$

Note that equation (6.110) establishes that the predictive mean, calculated on the average observations at replicates, and with covariances calculated only at the unique input locations, is indeed identical to the original predictive equations built by overlooking the structure of replication. Equation (6.111) reveals the same result for the predictive variance. Similarly, applying the Woodbury identity (6.109) in the inference equation yields

$$\begin{aligned}\mathcal{L} = &-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{m})^T\boldsymbol{R}_N^{-1}(\boldsymbol{y}-\boldsymbol{m}) + \frac{1}{2}(\hat{\boldsymbol{y}}-\hat{\boldsymbol{m}})^T\boldsymbol{A}_n\boldsymbol{R}_n^{-1}(\hat{\boldsymbol{y}}-\hat{\boldsymbol{m}}) - \\ &-\frac{1}{2}(\hat{\boldsymbol{y}}-\hat{\boldsymbol{m}})^T\left(\boldsymbol{K}_n+\boldsymbol{A}_n^{-1}\boldsymbol{R}_n\right)^{-1}(\hat{\boldsymbol{y}}-\hat{\boldsymbol{m}}) - \frac{1}{2}\log\left|\boldsymbol{K}_n+\boldsymbol{A}_n^{-1}\boldsymbol{R}_n\right| + \\ &+\log|\boldsymbol{R}_N| + \log\left|\boldsymbol{A}_n^{-1}\boldsymbol{R}_n\right|\end{aligned} \tag{6.112}$$

We can see here the reason for the requirement of a diagonal $\boldsymbol{R}_N$, otherwise, the complexity of the computation of the log-likelihood would remain the same as in the standard Gaussian Process. Then, by exploiting the structure of replications, we are able to retrieve the exact same model, reducing the computational complexity of the prediction and inference equations from $\mathcal{O}(N^3)$ to $\mathcal{O}(n^3)$.

In order to illustrate the potential of exploiting the structure of replications in the Gaussian Process design, consider a manipulation task modeled by

$$y(x) = 5\sin\left(\frac{4\pi}{10}x - \frac{\pi}{4}\right)\frac{\exp(-0.3x)}{1+\exp(-0.5x)} + \mathcal{N}\left(0\,,\ \frac{0.5}{1+\exp\left(\frac{-x+5}{2}\right)}\right) \tag{6.113}$$

where the second term represents the input-dependent uncertainty.

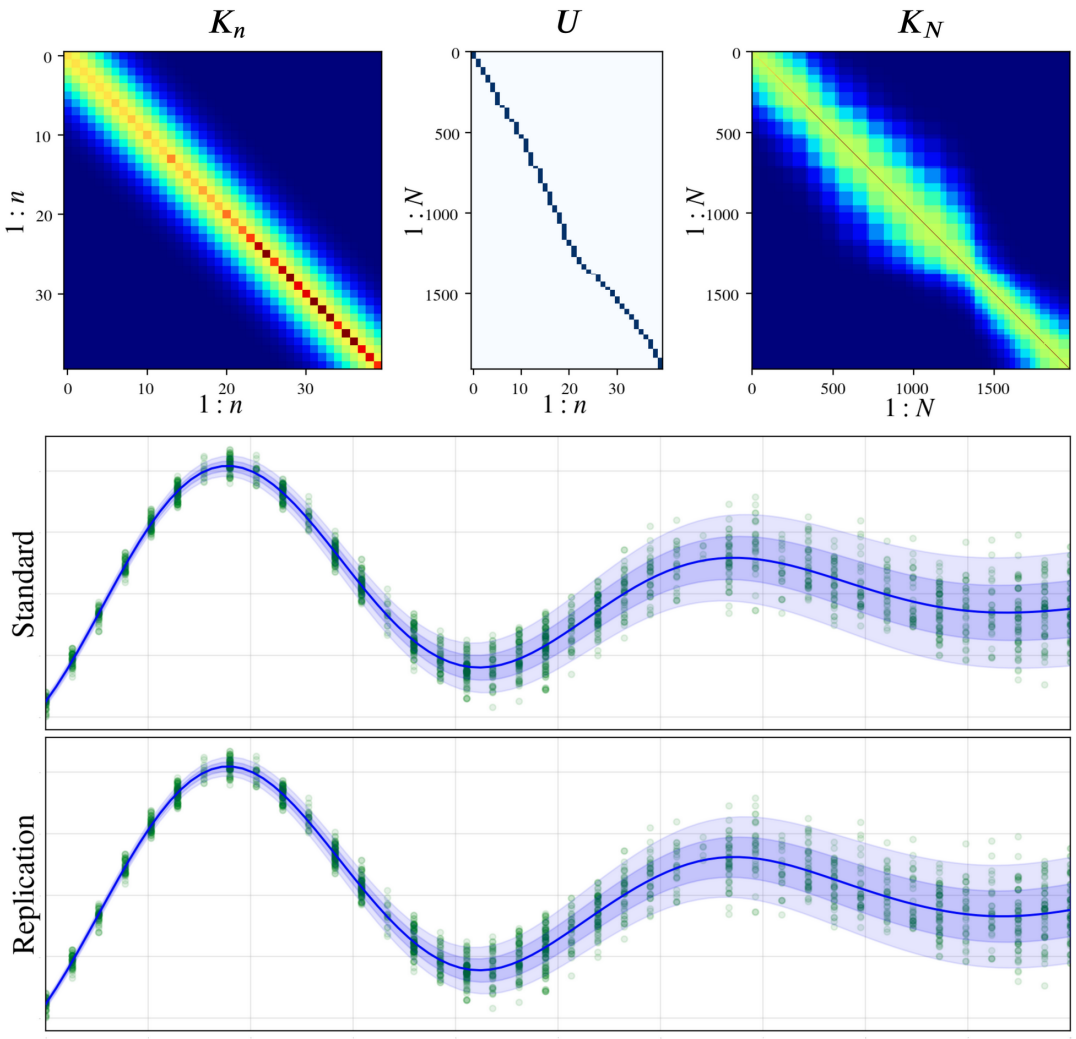

Figure 6.24: On the top row, the mapping between the full covariance matrix $\boldsymbol{K}_N$ and its unique-n counterpart $\boldsymbol{K}_n$ through the replicate decomposition $\boldsymbol{K}_n \to \boldsymbol{U} \to \boldsymbol{K}_N$ (6.106). On the second and third rows, the resulting Gaussian Process model after the inference and prediction steps, overlooking and exploiting the structure of replications, respectively. The green dots depict the training samples. The computation time required for the former is $t_s = 0.2284$s, and for the latter, $t_{rep} = 0.0021$s.

In Figure 6.24, we can see the results after training a Gaussian Process model on $N = 1969$ samples from (6.113). The demonstration dataset presents a high degree of replication since we have only $n = 40$ unique input locations. We can see that the covariance matrix $\boldsymbol{K}_n$ is similar to $\boldsymbol{K}_N$ but with a size considerably smaller. We can also observe that the model retrieved overlooking the structure of replications is identical to the one obtained considering the replicate decomposition. This is due to the exact nature of the method. The main difference lies on the computation time required for inference and prediction. For the standard model it takes $t_s = 0.2284$s while for the replication-based model it takes just $t_{rep} = 0.0021$s.

6.6 Application Examples

Endowed with higher levels of autonomy, robots are required to perform increasingly complex manipulation tasks. Learning from demonstration is a promising paradigm in this direction since it allows robots to learn tasks that cannot be easily scripted, but can be demonstrated by a human teacher. In this section, we illustrate the potential of the Gaussian-Process-based framework presented in this chapter through a series of real-world experiments. First, in Section 6.6.1, we address the problem of learning to open doors [90]. Then, in Section 6.6.2, we evaluate some of the most relevant aspects of our method through the robot writing task [91].

6.6.1 Door Opening Task

Service robots operating in domestic environments are typically faced with a variety of objects they have to deal with or they have to manipulate to fulfill their task. A further complicating factor is that many of the relevant objects are articulated, such as doors. The ability to deal with such articulated objects is relevant for assistive robots, as, for example, they need to open doors when navigating between rooms and to open cabinets to pick up objects in fetch-and-carry applications [92].

Opening doors is a task that is intuitive to teach but is difficult to hard-code since they are objects that come in a wide variety of sizes. For example, consider a room door, a refrigerator door, a small cabinet door, etc. We distinguish between two parts in the learning scheme: first, the inference of the policy from the demonstrations of the human teacher, which requires an adequate Gaussian Process design; second, the online adaptation to the current task requirements during the execution of the door opening motion based on the robot's sensorial feedback.

Policy Inference from Human Demonstrations

Demonstrations are recorded using the Xsens MVN motion capture system (a brief description of the device is given in Section 5.4.2). In Figure 6.25, we show some pictures of the human teacher performing the door opening motion. We can see that the interface for providing the demonstrations is basically a gown with wearable sensors that tracks several limbs of the operator.

We record the position and orientation of the operator's right hand relative to the initial position with the door closed. The reference frame is selected such as that the initial pulling direction is parallel to the x-axis and the y-axis is perpendicular to the floor. For the demonstration dataset, we consider three different doors and two trajectories per each door i.e., a total of six trajectories.

Figure 6.25: Demonstrations of the door opening motion are recorded using the MVN motion capture system. The human teacher opens doors with different radius.

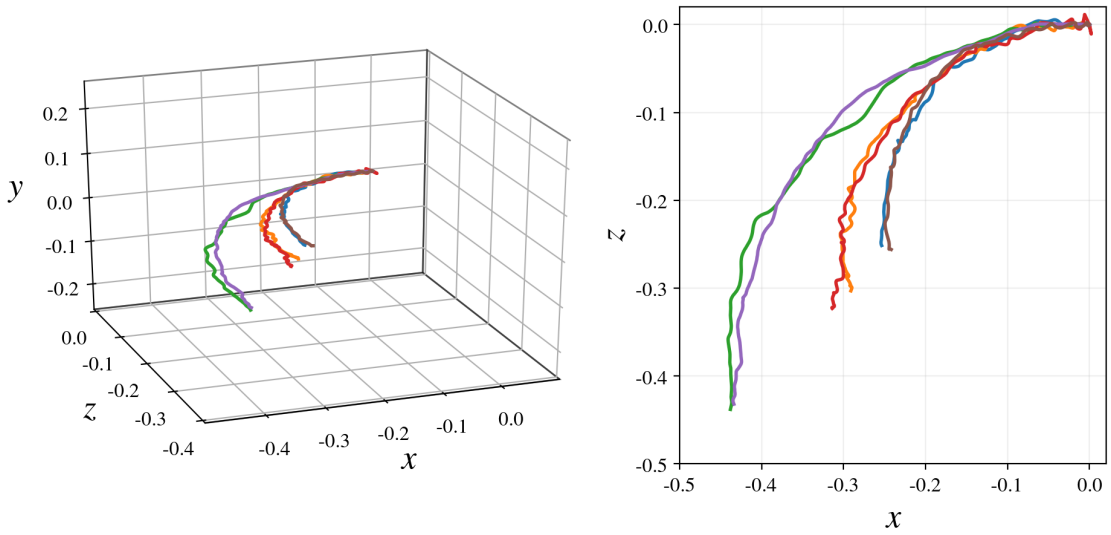

Figure 6.26: Recorded right-hand trajectories of the demonstrations dataset.

The demonstration dataset is shown in Figure 6.26. We can see that the door opening motion is planar. Thus, we can encode the trajectory using three coordinates, two for the position and one for the orientation. For the policy input, we consider only a phase variable for capturing the sequential aspects of the task. With respect to the selection of the kernel, the generated paths have to be continuous and smooth. Also, we have to take into account that the proposed phase parametrization of the trajectory is invariant to translations in the input domain. Then, the covariance function must be stationary. All these requirements are fulfilled by the squared exponential kernel (6.16). Moreover, since we have multiple outputs, we also have to consider the prior interaction in the design. Since in a general case we usually do not have any previous knowledge, we assume that the output components are independent.

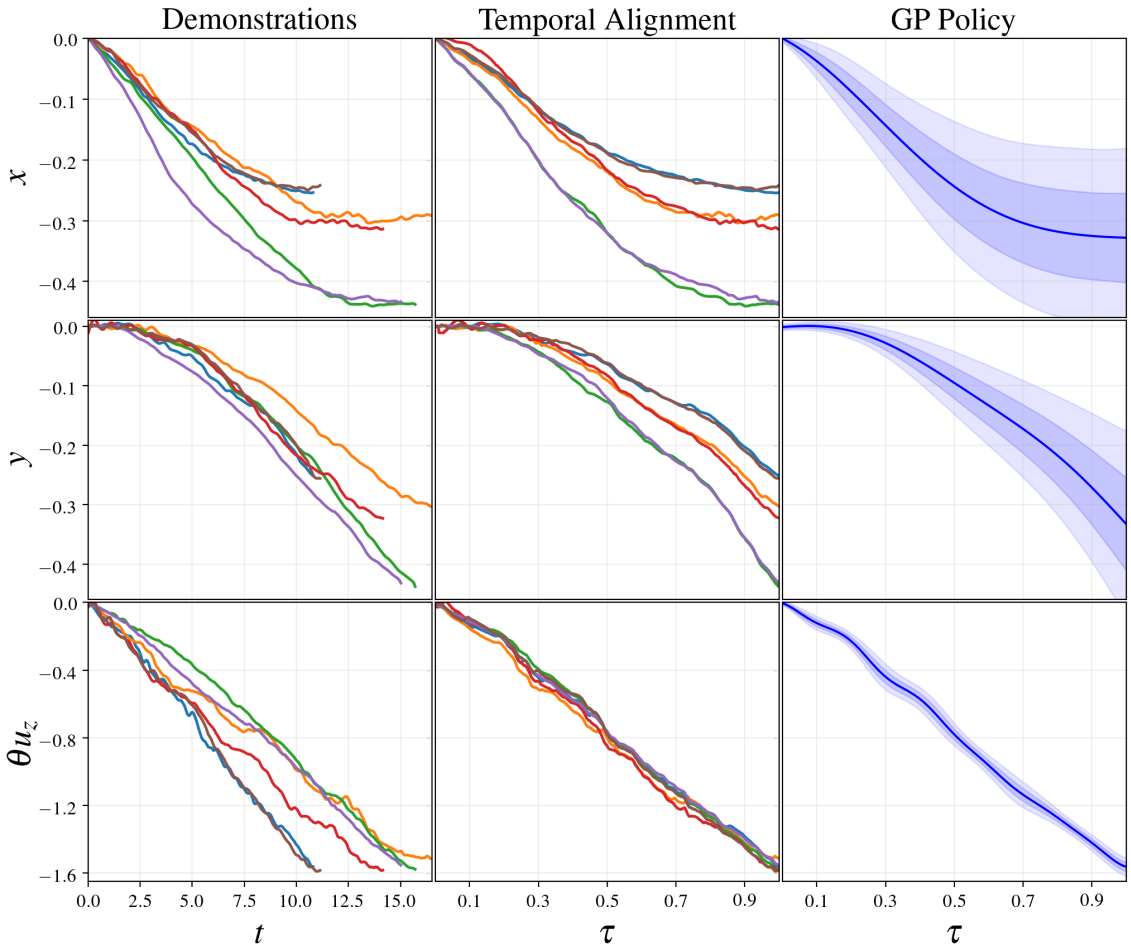

Figure 6.27: Inference of the door opening policy from human demonstrations. The outputs are the position, defined by (x, z), and the orientation defined by θu_z taking the axis-angle representation. The input is a phase variable τ. On the left column, we have the demonstrations of the door opening motion. On the middle column, the trajectories temporally aligned. On the right column, the inferred Gaussian Process (GP) policy.

The main steps of the learning process of the door opening policy are illustrated in Figure 6.27. The rotation component is encoded using the axis-angle representation. The demonstrated trajectories are aligned with the Dynamic Time Warping algorithm using the task completion index. We can see that the trajectories are warped effectively since they are clearly clustered in three different groups, one for each type of door. Once the trajectories are aligned, we infer the task policy training a heteroscedastic Gaussian Process model on the demonstration data. We can observe that the model effectively captures the door opening skill. This is more clear in Figure 6.28, where the task uncertainty has been projected onto the x-z plane. In this case, the variability in the task comes from the uncertainty in the radius of the door, which is reflected in the resulting policy.

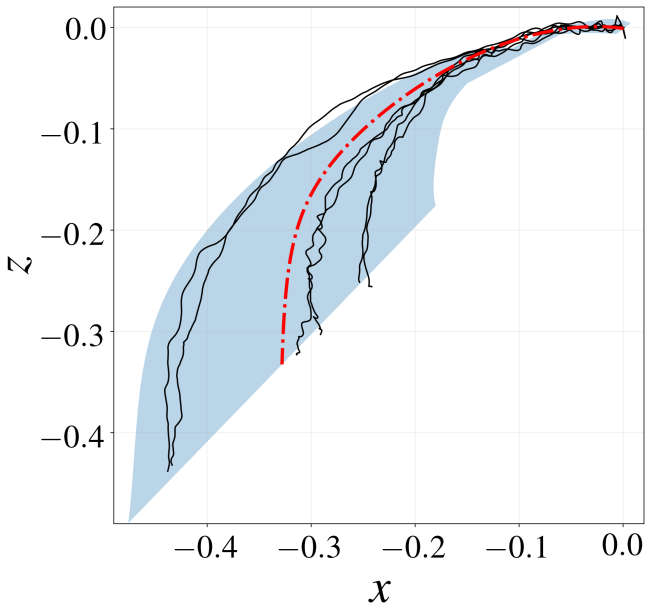

Figure 6.28: Door opening policy in Figure 6.27, projected onto the x-z plane.

Policy Adaptation During the Execution of the Task

During the execution of the task, we can exploit the observations of the motion of the door which is currently being opened to adapt the learned policy. Specifically, we can gather these data by solving the forward kinematics of the robot, and use it to define a set of via-point constraints. By updating this set at each time step we can adapt the motion online to the current task requirements. In order to evaluate quantitatively the performance of the adaptive policy against the one based solely on the demonstrations, we use the mean squared prediction error (MSPE). Assuming that there exists a ground truth policy $\widetilde{\pi}()$, which is the case when opening a door, the MSPE summarizes the predictive ability of the model. Ideally, this value should be close to zero

$$\mathbb{E}\left[\varepsilon^2\right] = \left(\mathbb{E}\left[\pi(X_*)\right] - \widetilde{\pi}(X_*)\right)^2 + \mathbb{V}\left[\pi(X_*)\right] \tag{6.114}$$

where $\mathbb{E}[]$ and $\mathbb{V}[]$ refer to the expectancy and the variance, respectively. The evolution of the adaptive policy and the MSPE during the execution of the door opening motion is shown in Figure 6.29. We can see that by conditioning on the current observations of the door we are able to reduce the task uncertainty in the near future, converging also the mean to the ground truth. This translates into better performance in terms of the MSPE, as we can see in Figure 6.29b). It is reduced by almost two orders of magnitude in the final stages of the task. With the proposed approach we are able to successfully perform the door opening task with the TIAGo robot, as can be seen in Figure 6.30.

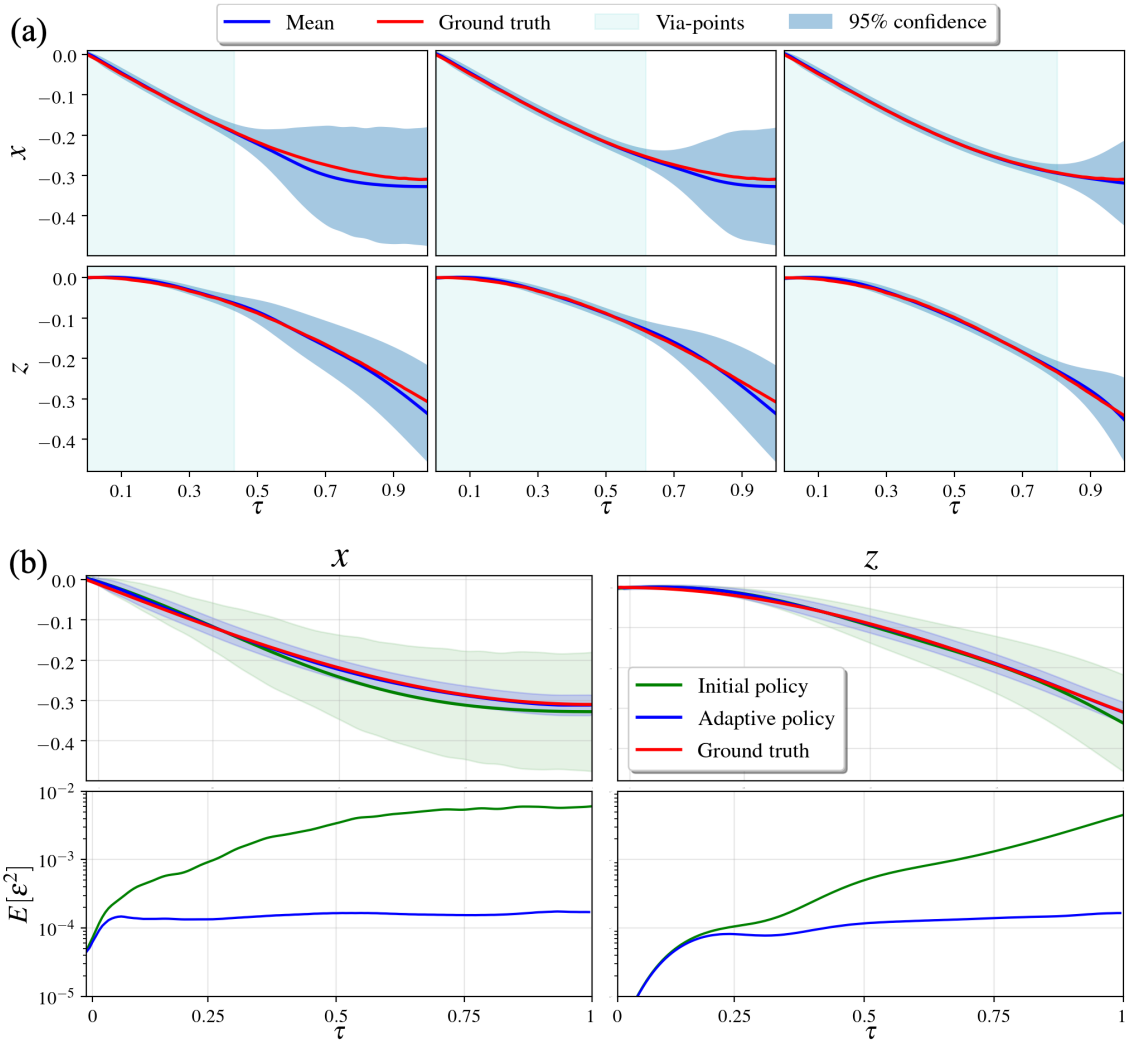

Figure 6.29: (a) Evolution of the posterior predictive distribution considering as via-points the observations of the door motion in the light-blue shaded area. (b) The first row shows the comparison between the predictive distribution considering the adaptive policy or the policy based only on human demonstrations. In the second row we can see the mean squared prediction error (6.114) of each policy.

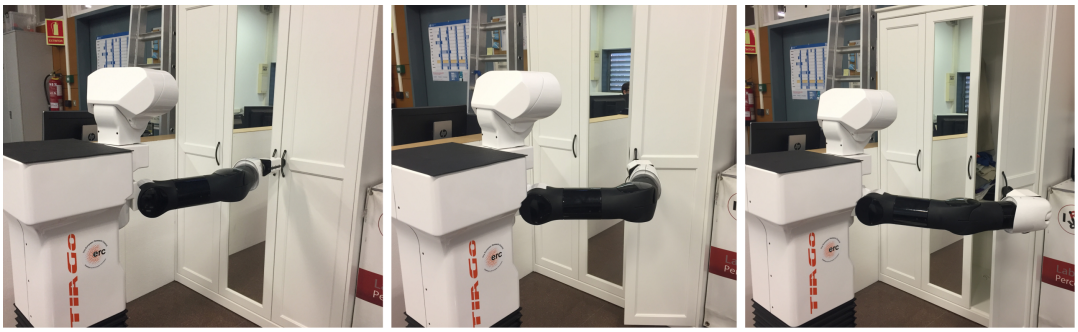

Figure 6.30: TIAGo robot opening the door.

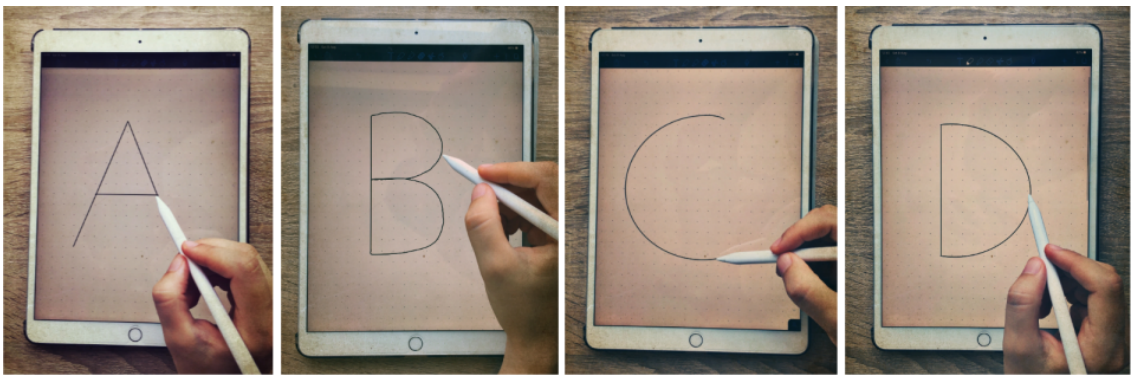

Figure 6.31: Demonstrations of the handwriting task are performed using a tablet.

6.6.2 Learning to Write

Learning to write is a skill that suits perfectly in the learning from demonstration context. The task has been explored for engaging robots in teaching activities. By letting a child teach a robot how to write, not only does the child practice handwriting but, as they take on the role of the teacher, they also positively reinforce their self-esteem and motivation [93].

Through the writing task, we illustrate and evaluate some of the main concepts of the presented Gaussian-Process-based learning from demonstration framework. First, we build an experimental handwritten dataset, which includes trajectories for several letters indexed by real, integer, and categorical task variables. Then, we consider the problem of learning a policy from the demonstrations, comparing the performance of different Gaussian Process designs. Next, we assess the adaptability of the resulting policy by evaluating the modulation through via-points, and the interpolation and extrapolation capabilities. Finally, we study, in terms of computational time, the implications of considering the structure of replications.

Handwritten Letter Dataset

The demonstration dataset has been generated experimentally by handwriting different letters on a tablet using a standard note app (Figure 6.31). We extracted the data by first screen recording while writing, and then, processing the resulting videos with computer vision techniques. As output variables, we take the x and y coordinates of the path that describes the handwriting motion. As input variables we take a phase variable ρ; the size of the letter ι, which we consider defined by the height, measured as an integer height $= \iota \times 8$mm; and finally, the letter corresponding to the motion κ, e.g. 'A', 'B', 'C', etc. The variables and the domain sampled in the dataset are summarized in Table 6.1.

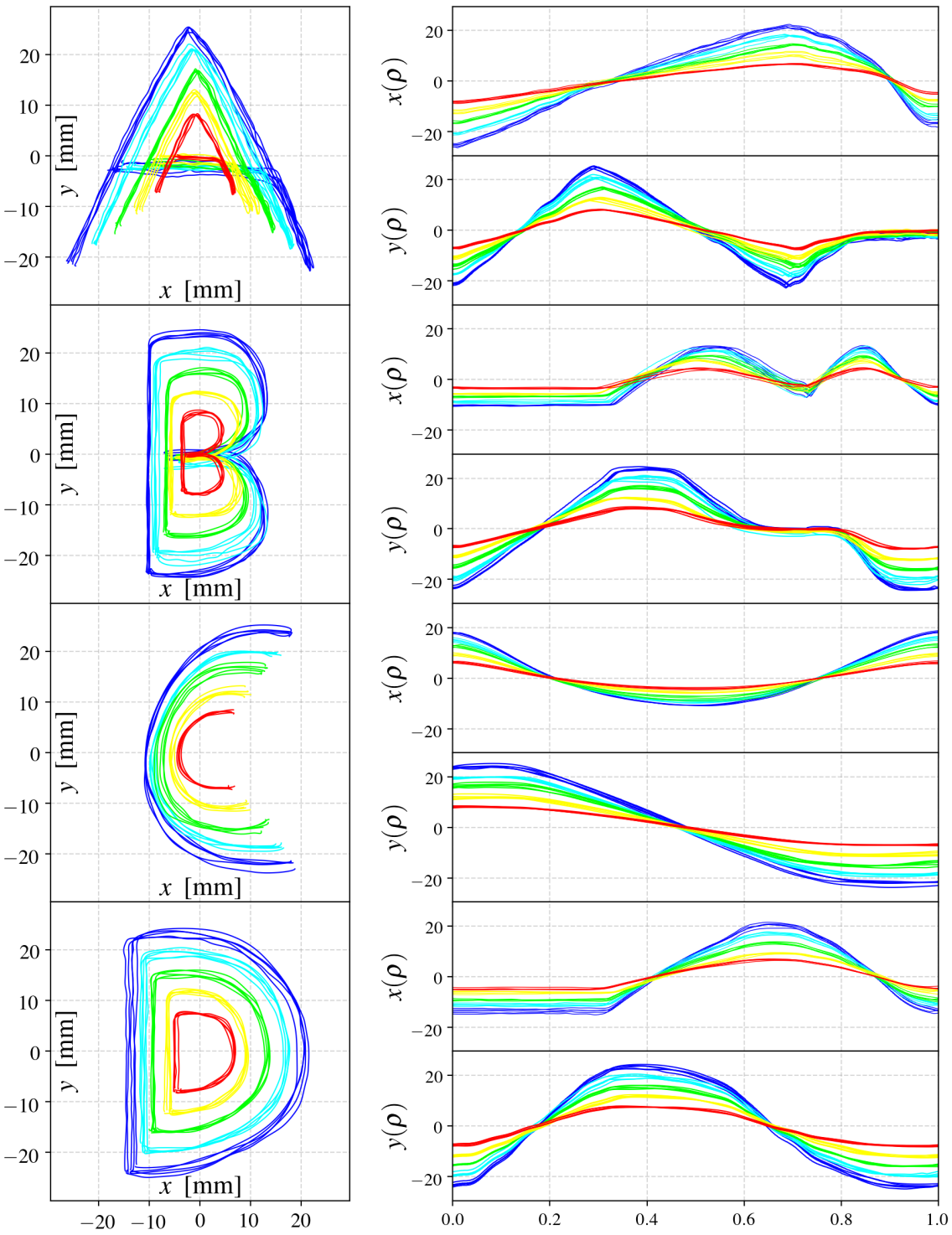

Figure 6.32: Handwritten letter dataset. Trajectories are indexed by the input variables indicated in Table 6.1. A total of 5 replicates are provided for each possible combination of the discrete variables ι and κ, represented in the same color. On the left column, each pair of trajectories $x(\rho)$, $y(\rho)$ on the right column is projected onto the x-y plane.

Variable	Symbol	Type	Domain
Time	ρ	Input	$[0, 1]$
Size	ι	Input	$\{2, 3, 4, 5, 6\}$
Letter	κ	Input	$\{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$
Horizontal coordinate	x	Output	$\mathbb{R}$
Vertical coordinate	y	Output	$\mathbb{R}$

Table 6.1: Handwritten letter dataset variables.

The size of the dataset was guided by the discrete task variables, such that each of the $5 \times 4 = 20$ possible discrete input locations have 5 replicates. That is a total of 100 different demonstrations. The dataset, after aligning temporally the trajectories, is shown in Figure 6.32.

Task Policy Design

We aim to learn a movement policy from the demonstrations that is capable of generating the motion required to write a given letter with a specified size. As in the presented Gaussian-Process-based approach, we focus the modeling effort on the covariance structure. A-priori, the outputs are uncorrelated since the relation between them depends on the letter being written. Thus, we use a diagonal matrix covariance function for the multi-output Gaussian Process.

Regarding the diagonal elements, since we have multiple input dimensions of different nature, we can build the kernel as a composition, which can be either sum (6.73), product (6.74) or ANOVA (6.75), of one-dimensional kernels. For $k_{\mathbb{R}}$ we consider the squared-exponential and the Matérn; for $k_{\mathbb{Z}}$ we propose the cosine kernel (6.68); finally, for $k_{\mathbb{K}}$ we take the CS structure (6.69). To select the best model, we split the dataset in 50% for learning the task policy and the remaining 50% for assessing its performance. The results obtained for different covariance structures in terms of the coefficient of determination R^2 coefficient are shown in Table 6.2. Although very accurate predictions are obtained either with the product or ANOVA composition, the best score is achieved by the ANOVA+Matérn model. In Figure 6.33, we can see that the task policy effectively retrieves the motion required for different letters and sizes, also encoding the uncertainty.

Composition	Product	Sum	ANOVA
$k_{\mathbb{R}} =$ **SE**	0.87	0.32	0.93
$k_{\mathbb{R}} =$ **Matérn**	0.89	0.35	0.94

Table 6.2: Coefficient of determination R^2 for different covariance structures.

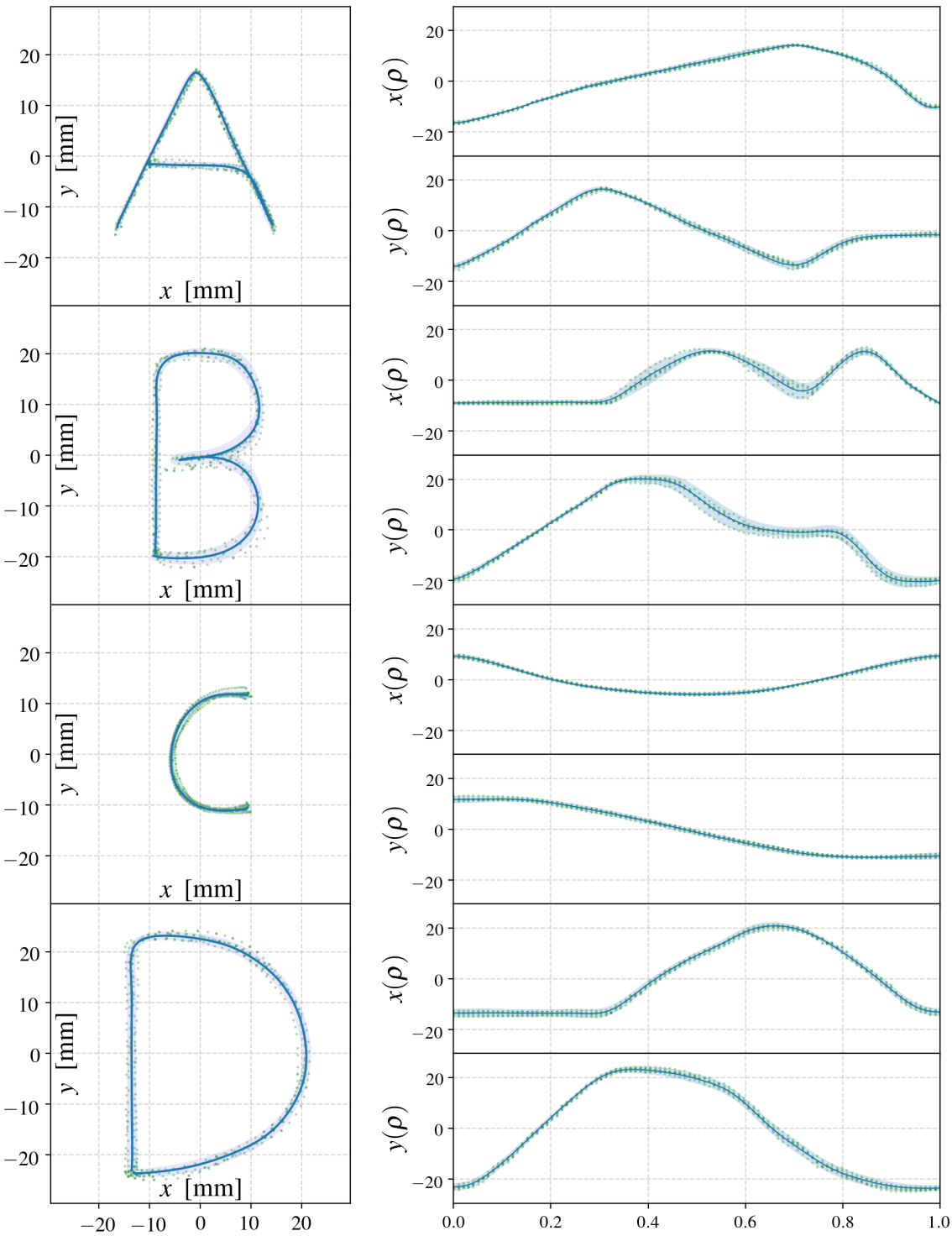

Figure 6.33: Motion retrieved by the task policy learned from the demonstrations for writing different letters with different sizes. The solid blue line denotes the mean, the blue shaded area, the 95% confidence interval, and the green dots, the training data.

Generalization Capabilities of the Task Policy

Here, we study the generalization capabilities of the learned task policy by evaluating how is it adapted to unseen scenarios. This adaptation can be achieved either through the specification of via-points, or relying on the GP model to generate the required motion given a new set of task parameters, for which demonstrations are not provided. This requires interpolation or extrapolation from the training dataset.

The obtained results are shown in Figure 6.34. In Figure 6.34a), we can see that the motion is easily modulated to fulfill new specifications by conditioning the learned distribution to pass through new initial, final and/or intermediate via-points. In case that during the execution of the task the robot encounters a new context, not sampled in the demonstration set, we consider two different scenarios. In Figure 6.34b), we only use data of letters with sizes 3 and 5. Thus, for writing a letter 'A' of size 4, interpolation is required. We can see that the policy is capable of predicting accurately the trajectory for generating the required motion with relatively low uncertainty. On the other hand, in Figure 6.34c), we only consider demonstrations for sizes 5 and 6, requiring extrapolation for writing a letter 'A' of size 4. Since the unseen task variables are close to the demonstration region, we can see that the policy is capable of adapting effectively. Note that the task uncertainty is higher than in the case of interpolating.

Computational Advantage of Exploiting the Structure of Replications

Now, we evaluate the potential benefit of exploiting the structure of replications during the inference and prediction steps of the Gaussian-Process-based policy. From each of the 100 demonstrations in the dataset, we take 25 uniformly distributed samples along the real dimension for all possible combinations of the task variables for training. Thus, we have a total of $N = 2500$ points with $n = 500$ unique input locations, that is, we have a degree of replication of $a = 5$. In Figure 6.35, we show a comparison of the computational time per evaluation of the inference function (inference), and for calculating the posterior predictive distribution of the policy (prediction). We can see that we are able to perform the calculations 100 times faster without any approximation, achieving identical results.

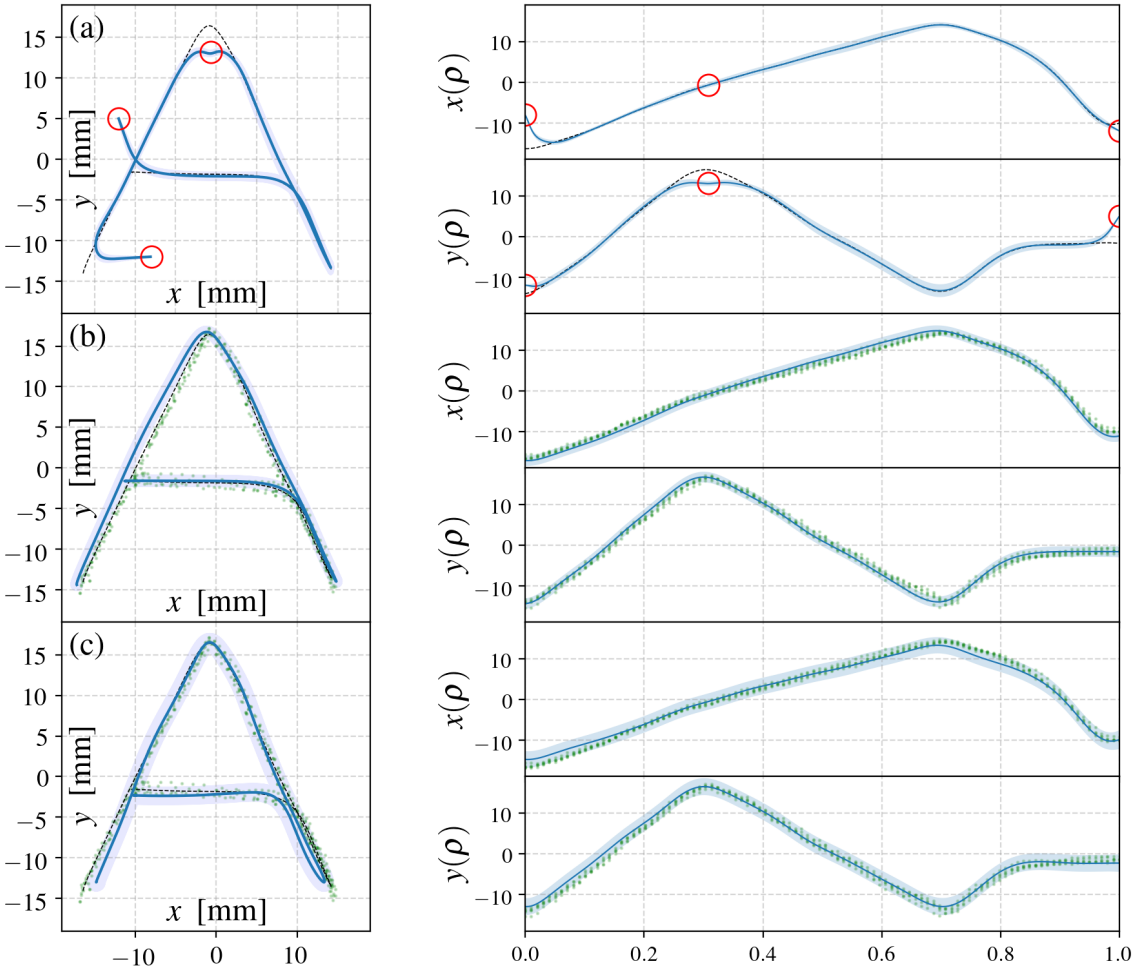

Figure 6.34: (a) Policy adaptation through via-points (red circles). For comparison, the mean of the demonstrations is represented as a black dashed line. (b) The model is built only with demonstrations of sizes 3 and 5, requiring interpolation for size 4. (c) In this case, only demonstrations of sizes 5 and 6 are used, thus extrapolation is performed.

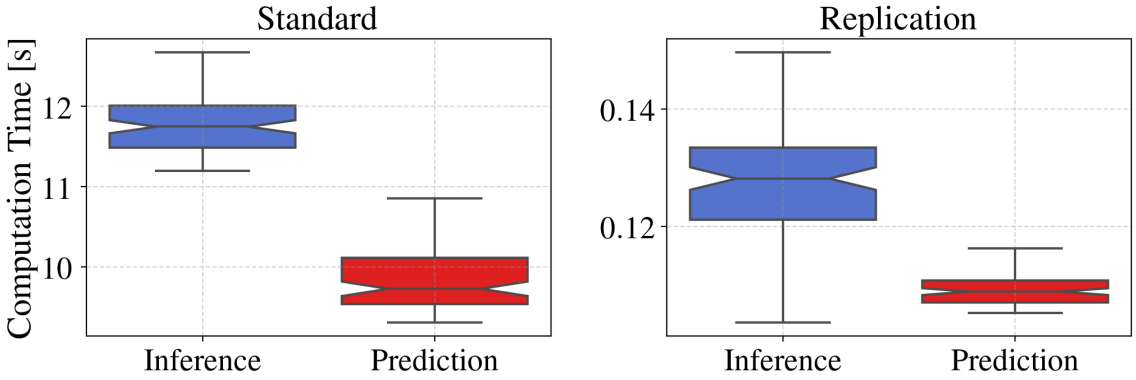

Figure 6.35: Computational advantage of exploiting replications for evaluating the hyperparameter inference function, and calculating the predictive distribution.

Part III

Robots and Society

Chapter 7

Economic Impact of Robotics

Rapid advances in technology have led to a surge of public interest in robotics. The International Federation of Robotics (IFR), estimates that by the end of 2021 around 730,000 service robots will be used worldwide in the industrial sector and around 50 million service robots in the end consumer sector [94]. This forecast shows that service robots have the potential to become part of the professional and private lives of many people in the future. Furthermore, some researchers anticipate that this growth will be boosted by the global impact of the COVID-19 pandemic due to the necessity of measures for physical distancing and isolation [95].

The rapid development of the robotics market has sparked a lively debate among economists, journalists, and technophiles about the likely impact of automation on economic growth and the distribution of income. Mainly, two narratives have emerged in the literature [96]. On the one hand, technology pessimists fear that we are headed toward an economic dystopia of extreme inequality and class conflict [97]. On the other hand, technology optimists argue that historically, rapid technological advancements have led to an improvement in human welfare by raising the value of non-automatable tasks [98].

Automation is not a new phenomenon, and questions about its effects have long accompanied its advances. More than a half-century ago, US President Lyndon B. Johnson established a national commission to examine the impact of technology on the economy and employment, declaring that automation did not have to destroy jobs but 'can be the ally of our prosperity if we will just look ahead' [99]. In this chapter, we aim to shed some light on the debate around the potential economic impact caused by the introduction of robots in our daily lives (Figure 7.1). We start, in Section 7.1, discussing the implications of an automated society from a macroeconomic perspective. Then, in Section 7.2, we study how robotics will change the labor market. Finally, in Section 7.3, we present some guidelines for addressing some of the new challenges that we might face in the future robot economy [100].

Figure 7.1: Illustration of an automated economy (after [101]).

7.1 Macroeconomic Impact

Although economists have extensively analyzed the effects of technological change, the formal theoretical macroeconomic literature that bears on the current debate is quite small. Predicting the economic impact of robotics or any disruptive technology is a very complex exercise [102]. This is a world of near-continuous discontinuity. The scope and pace of automation deployment depend on several variables (some more predictable than others) including technical feasibility, the cost of developing and deploying technologies for specific uses in the workplace, labor-market dynamics including the quality and quantity of labor and associated wages, the benefits of automation beyond labor substitution, and regulatory and social acceptance.

The key idea behind automation from a macroeconomic perspective is that it improves productivity when it is applied to tasks that they perform more efficiently and to a higher and more consistent level of quality than humans. Gross Domestic Product (GDP) growth was brisk over the past half-century, driven by the twin engines of employment growth and rising productivity, both contributing approximately the same amount. However, declining birth rates and the trend toward aging in many advanced and some emerging economies mean that peak employment will occur in most countries within 50 years. The expected decline in the share of the working-age population will open an economic growth gap: roughly half of the sources of economic growth from the past half-century (employment growth) will evaporate as populations age.

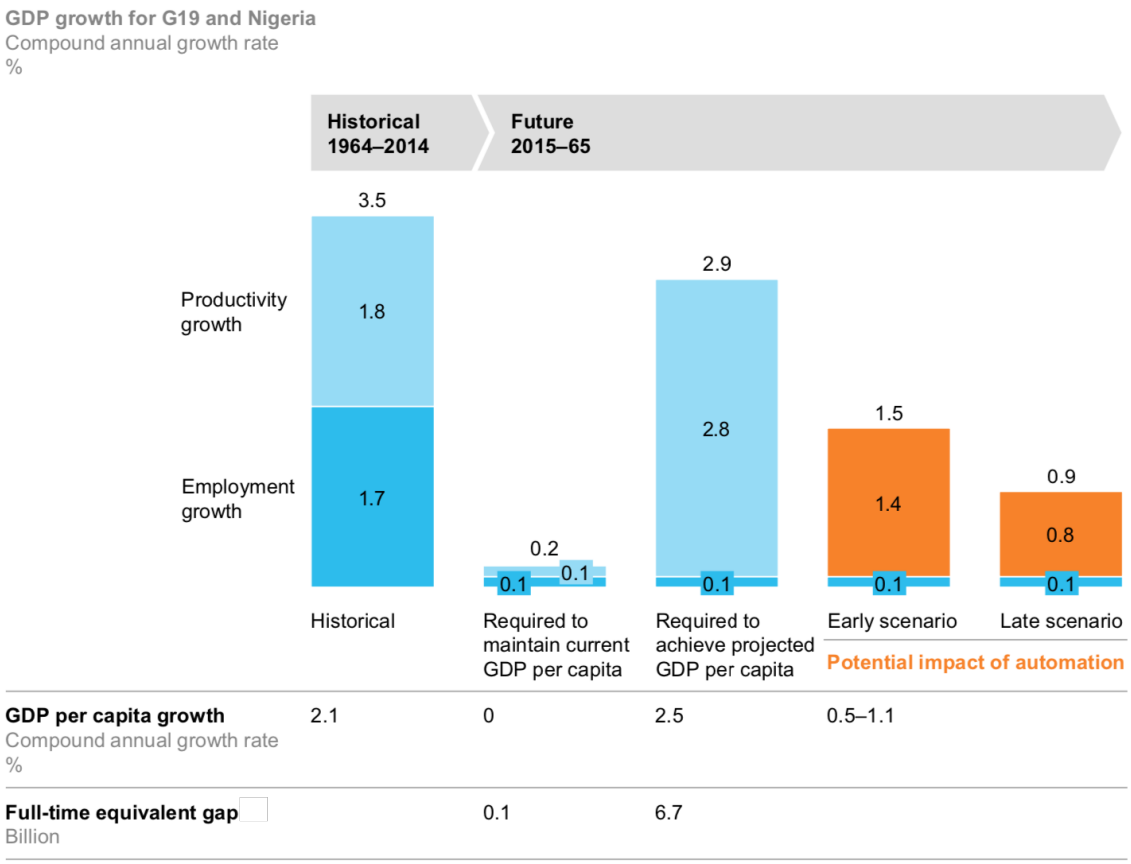

Figure 7.2: Globally, automation could become a significant economic growth engine as employment growth wanes (after [99]).

Even at historical rates of productivity growth, economic growth could be nearly halved. Automation could compensate for at least some of these demographic trends. McKinsey [99] estimates that the productivity injection that automation could give to the global economy as being between 0.8 and 1.4 percent of global GDP annually. By 2065, automation could potentially add productivity growth in the largest economies in the world (G19 plus Nigeria) that is the equivalent of an additional 1.1 billion to 2.3 billion full-time workers (Figure 7.2).

Increased productivity is enabling some firms (such as Whirlpool, Caterpillar, and Ford Motor Company in the US and Adidas in Germany) to restructure their supply chains, bringing back parts of the manufacturing process to the country of origin. Companies that deploy robots are less likely to relocate or offshore in the first place according to a report prepared for the European Commission [103]. This brings advantages at the national level, with the potential for demand spillovers into other sectors, and the accumulation of specialist manufacturing know-how that is critical for attracting and expanding talent, and for national competitiveness [94].

The productivity growth enabled by automation can ensure continued prosperity in aging nations and provide an additional boost to fast-growing ones. This could help close the economic growth gap in the 20 largest economies in the medium term, to 2030. Countries can use automation to further national economic growth objectives, depending on their demographic trends and growth aspirations. We can divide the world countries into three main groups [99], based on how can they exploit robotics to boost their economic growth:

- **Advanced economies**: Including Europe, the United States, Canada, Australia, Japan, and South Korea. These economies typically face an aging workforce, with the decline in the working-age population. Automation can provide the productivity boost required to meet economic growth projections that they otherwise would struggle to attain without other significant productivity growth accelerators. These economies thus have a major interest in pursuing rapid automation adoption.
- **Emerging economies with aging populations**: This category includes Argentina, Brazil, China, and Russia, which face economic growth gaps as a result of projected declines in the growth of their working population. For these economies, automation can provide the productivity injection needed just to maintain the current GDP per capita. To achieve a faster growth trajectory that is more commensurate with their developmental aspirations, these countries would need to supplement automation with additional sources of productivity, such as process transformations, and would benefit from the more rapid adoption of automation.
- **Emerging economies with younger populations**: These include India, Indonesia, Mexico, Nigeria, Saudi Arabia, South Africa, and Turkey. The continued growth of the working-age population in these countries could support maintaining current GDP per capita. However, given their high growth aspirations, automation plus additional productivity-raising measures will be necessary to sustain their economic development.

Productivity gains due to robotics and automation are important not just at the company level but also for both industry and national competitiveness. However, we have also to take into account that the advances in automation and their potential impact on national economies could upend some prevailing models of development and challenge ideas about globalization. Countries experiencing population declines or stagnation will be able to maintain living standards even as the labor force wanes. Meanwhile, countries with high birth rates and significant growth in the working-age population may have to worry more about generating new jobs in a new age of automation. Moreover, low-cost labor may lose some of its edges as an essential developmental tool for emerging economies, as automation drives down the cost of manufacturing globally.

7.2 Labour Market Impact

With the raising of robotics and artificial intelligence, one of the most repeated questions in the mainstream media is: 'will robots take our jobs?'. What is already clear and certain is that new technological developments will have a fundamental impact on the global labor market within the next few years, not just on industrial jobs but on the core of human tasks in the service sector. Economic structures, working relationships, job profiles, and well-established working time and remuneration models will undergo major changes [104].

Both blue-collar and white-collar sectors will be affected. Nowadays, robots not only do things that we thought only humans could do but also can increasingly do them at superhuman levels of performance. Some robots are far more flexible (and a fraction of the cost) than those used in manufacturing environments. Today, robots can be 'trained' by frontline staff to perform tasks that were previously thought to be too difficult for machines, and are even starting to take over service activities.

In this section, we discuss how will the irruption of robotics is going to change the labor market. First, in Section 7.2.1, we study the impact on the job profile demand. Then, in Section 7.2.2, we analyze the effect on the organization of work. Finally, in Section 7.2.3, we argue how it will affect the time spent at work.

7.2.1 Impact on the Job Profiles Demand

A recent study by McKinsey [99] focuses on work activities rather than whole occupations. This is a more relevant and useful measure since occupations are made up of a range of activities with different potential for automation. For example, a retail salesperson will spend some time interacting with customers, stocking shelves, or ringing up sales. Each of these activities is distinct and requires different skills.

Overall, it is estimated that 49 percent of the activities that people are paid to do in the global economy have the potential to be automated by adapting currently demonstrated technology. While less than 5 percent of occupations can be fully automated, about 60 percent have at least 30 percent of activities that can technically be automated. While certain categories of activity, such as processing or collecting data, or performing physical activities and operating machinery in a predictable environment, have a high technical potential for automation, the susceptibility is significantly lower for other activities including interfacing with stakeholders, applying expertise to decision making, planning, and creative tasks, or managing and developing people. Figure 7.3 shows a range of sectors in the economy broken down into different categories of work activity along with their automation potential.

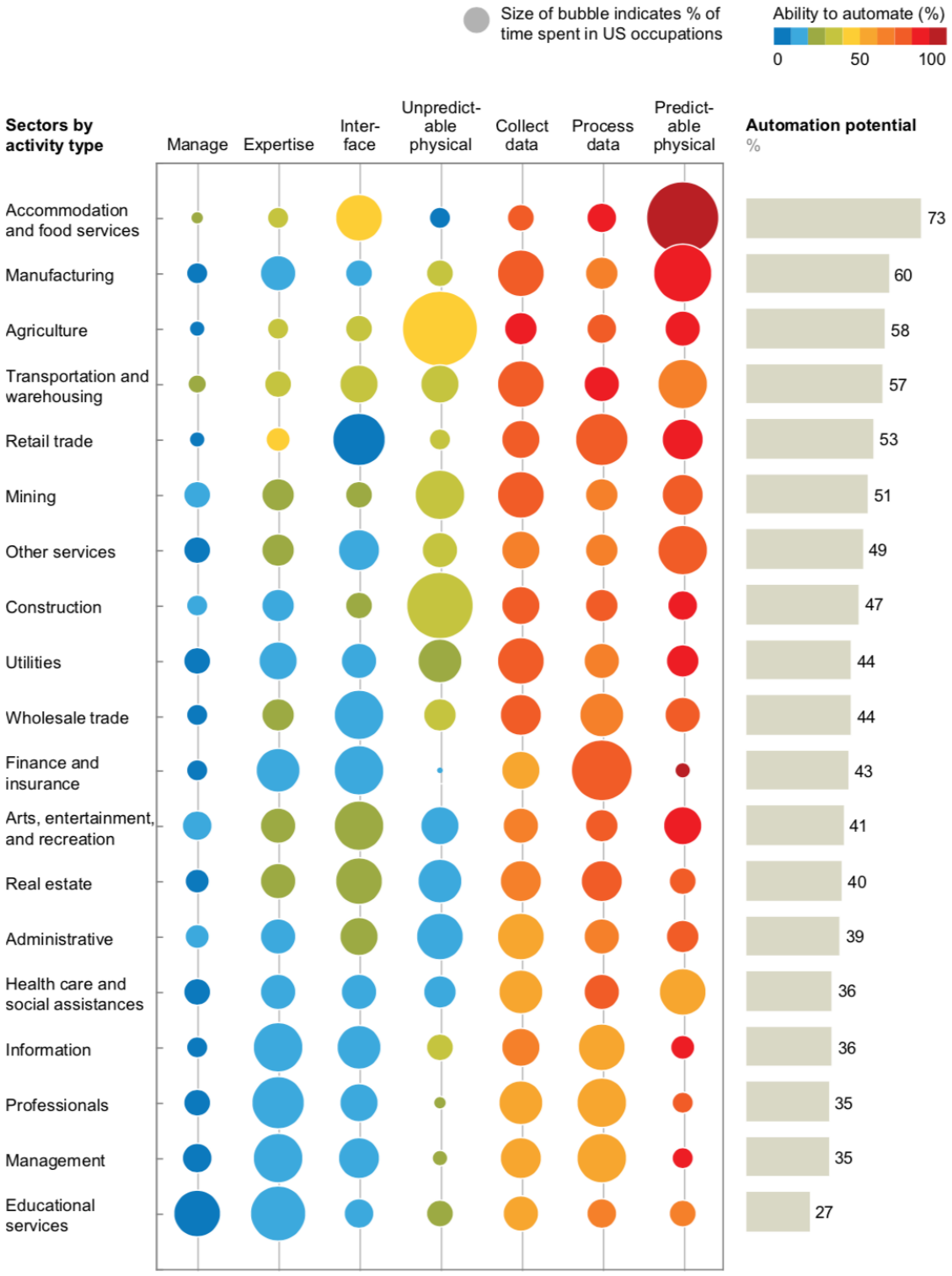

Figure 7.3: Technical potential for automation across sectors varies depending on the mix of activity types (after [99]).

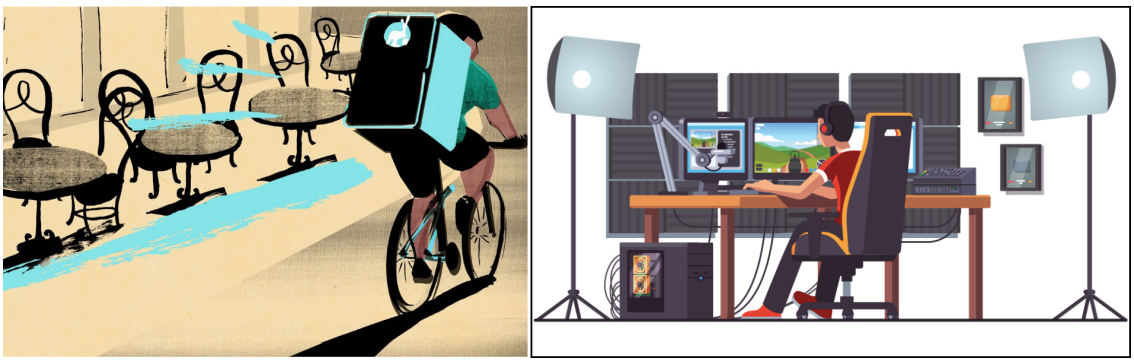

Figure 7.4: Zero-sum activities that have emerged in the twenty-first century.

Other potential impacts in the job profile demand can be explained by Baumol's cost disease [105]. Baumol observed that sectors with rapid productivity growth, such as manufacturing, often see their share of GDP decline while those sectors with relatively slow productivity growth, such as the service sector, experience increases. Rapid technological growth can be and often is accompanied by a proliferation of low productivity jobs taken up by workers no longer required in increasingly automated sectors. Thus, paradoxically, radical automation may eventually reduce towards zero the number of people involved in those economic activities essential to produce the goods and services which support human welfare, while having no such impact on the amount of human labor devoted to zero-sum activities (Figure 7.4), which may therefore grow in relative importance as we 'find things to do' [106]. In conclusion, robots will make work more human, demanding more creative skills rather than repetitive physical skills.

7.2.2 Impact on the Organization of Work

For economic reasons, numerous jobs will be carried out by intelligent software or machines rather than by humans in the future. The differentiation made in this regard will not be so much between physical and cognitive work, but primarily between routine and non-routine work. The established companies will need to create new business structures.

In the future, robotics engineers will be indispensable for many of the traditional departments in which a company is divided, such as sales, supply chain, production, and finance. Therefore, an interface must be established that ensures that information is passed on without delay to the offices responsible internally. But it is not only the departments that will have to be better connected. Companies will have to focus on their core competencies and will outsource other work in a cost-effective manner [104].

The increased interconnection and internationalization of companies changes not only the traditional internal company structures but also the need for the establishment of cross-company and cross-border units. Many companies already use so-called matrix structures today. They are characterized by the technical supervision and the disciplinary supervision of the employee being treated separately. Increasing digitalization makes this possible. Owing to the interconnection of individual companies, such unitary structures usually result in increased productivity due to improved communication and exchange of knowledge.

7.2.3 Impact on Working Time

Working time rules have been established in various countries in order to protect the health and safety of employees or to warrant pay for overtime work. One of the things that all of the systems have in common is that they are faced with fundamental reforms within the framework of automation.

Working life has to date been characterized worldwide by rigid standard working hours: employees were present in the establishment or in the office and worked there for a certain number of hours. The working day ended when the employees left the establishment after eight or ten hours of work. Nowadays, employees sometimes take their work home with them in the form of smartphones, laptops, and email. Critics claim that this makes it impossible to 'disconnect' from work, which damages employees' health over the long term. Additionally, self-determined working time can lead to higher motivation, increased health, and more productivity with regard to the individual employee.

A possibility may be the six-hour day in Sweden. Innovative employers report that shorter workdays improve the productivity of employees. New vistas are opened: thanks to digital innovations, families, especially, are able to adapt work and leisure time more flexibly to their own needs, which is said to lead to a better work-life balance. Supporters say this will cause employees to be more satisfied and motivated and will reduce stress and illness.

The new work might be geared more to the interests of the employees than before. The reason for this is that highly qualified staff can be 'winners', since research, development, design, planning, and organization jobs are now in demand, whereas purely productive occupations are on the decline since they are easily replaceable by robots. Companies must lure these prospective employees, unlike 20 or 30 years ago, not only by offering money but also by offering attractive arrangements for a better work-life balance. The future world of work will require employees to be much more flexible. Ties to fixed places of work or to fixed working hours will be increasingly broken.

7.3 Managing the Economic Impact

Business leaders, policymakers, and workers everywhere face considerable challenges in capturing the full potential of automation's beneficial effect on the economy, even as they navigate the major uncertainties about the social and employment repercussions. There is a basic doubleness in economics that reflects a fundamental social tension: the microeconomic perspective of individual businesses is to minimize labor costs, while the macroeconomic perspective of society as a whole requires consumers to have some source of income and for most of us, that is work. How society balances these rival demands of labor and capital is a subject of both politics and economics [107].

Thus, socio-economic, political, and resource constraints should be carefully considered when advanced robotic technologies are deployed since there is potential for unintended consequences such as tilting economic and power structures to unduly benefit certain segments of the society resulting in new gaps and/or exacerbating existing inequities. The results of theoretical and empirical investigations differ regarding the effects of robotization and AI on productivity, employment, and wages (as examples of recent studies with conflicting results can be consulted [108, 109, 110]). Also, in order to assess the impact of robotic technology on inequality, we would consider aspects such as ownership structures and tax systems [111].

Philosophy of economics that better corresponds to a coming automation economy featuring social robotics is needed. This would need to include reconceptualizations of the core ideas and purpose of economics. While material goods attend to survival, social goods attend to thriving. The automation economy is concerned not just with human survival, but an improved quality of life such that humans can thrive, and it is in this domain that social robotics could feature prominently. Possibly even more than technological unemployment and the possibility of 'robots taking our jobs', income inequality is a threat to the future well-being of our economies. Among the proposed measures to address the challenges of the future robot economy, we can highlight the following:

- **Constant free life-long education**: This is the most obvious solution to technological unemployment. As robots are capable of taking over a growing number of tasks, humans have to focus on their comparative advantages, including creative and social skills.
- **Universal basic income (UBI)**: This is widely discussed as a solution to technological unemployment. Under the UBI scheme, every citizen of a country receives a fixed amount of money every month regardless of his/her employment status. All other social payments are ceased and replaced by UBI.

The main advantage of UBI is that it provides income for all people in a society and serves as a social safety net. On the other hand, UBI requires a lot of resources. Furthermore, it may suppress many people's stimuli to work and improve their skills, thus making them permanently unemployable. Moreover, if UBI is introduced in one or a few countries only without strict migration control, they will experience a huge influx of immigrants who would apply for citizenship in order to take advantage of UBI [112].

- **Robot-based taxation**: This is considered as one of the ways to finance the UBI. Essentially, under a robot tax scheme, every company that installs a robot must pay tax, and the proceeds are used to support the UBI of displaced human employees. While robot-based taxation sounds quite attractive, is difficult to implement. By decreasing the capital returns of robots, the taxation has potentially negative effects on growth. This is also becoming more complicated with globalization and increasing capital mobility [96].

- **Job guarantee program (JGP)**: This is a proposal by critics of UBI that have pursued a broader vision for social purposes. Rather than guaranteeing income, JGP would guarantee jobs in those necessary fields, as a means for the government to ensure access to universal basic services. The program is intended to 'hire off the bottom', i.e. those with fewer qualifications or skills who otherwise cannot find employment [113]. According to modern monetary theory (MMT), the JGP is financially feasible when a sovereign government's currency uses a floating exchange rate [114]. The sovereign government risks inflation if it creates too much of its own currency. But the hope behind MMT is that wise spending will increase the productive capacity of the economy, sparking innovation to assure more efficient use of resources and mobilizing now-unemployed labor [115]. In any case, maintaining full employment is one of the chief responsibilities of modern governments, and a direct employment guarantee can be better targeted to this purpose than discretionary policy [116].

- **Employee ownership of robots**: If intelligent robots are substitutes for labor, the key is who owns the robots. While the previous measures are intended to establish transfers from rich to poor to reduce income inequality, this aims to directly reallocate income from owners to workers [97].

Managed adequately, the development of robotics can contribute to the improvement of human well-being. The automation of global value chains contributes to the divorce of humans from their labor, from their capacity to earn wages for sustaining human life. However, policymakers must begin to consider alternative economic arrangements which do not rely solely on the wage-labor contract at the heart of capitalism. The final socio-economic objectives of the development of production with robots must be those contributing to the improvement of society.

Chapter 8

Robot Ethics and Law

Our social spaces will no longer be exclusively human. The interactions with robots are increasing, and we can expect that we will be coexisting with social robots in the future. These robots will be nested within our social spaces and work with us side by side [117]. Service robots operating among people and often in close interaction with them raises many questions concerning their influence on the future of society and the role this technology may play.

As robotics technology advances, ethical concerns become more pressing (Figure 8.1). Should robots be programmed to follow a code of ethics, if this is even possible? Are there risks in forming emotional bonds with robots? How might ethics change with robotics? [118]? The expectations and the fears about the future consequences of autonomous technologies are monopolizing the ethical debate. Several initiatives and projects are underway in an attempt to address such questions in contexts as diverse as the military, labor market, and educational fields [119]. Also, if the legal conditions do not impose adequate restrictions, human society can give their robots substantial autonomy in decision-making, thus permitting them to complete tasks that involve sophisticated judgments. Intelligent robots can learn through experience, modifying and devising new instructions in their own programs. Then, they can make decisions based on these self-modified or self-created instructions, which can be dangerous. Robot law is an emerging field that aims to build a new legal framework that satisfies the needs of automated society [120].

In this chapter, we address some of the most relevant questions in ethics and law caused by the probable introduction of robots in our society. We start, in Section 8.1, presenting a series of ethical problems posed by the advance of automation. Afterward, in Section 8.2, we discuss the potential impact on the current legal framework. Finally, in Section 8.3, we provide some guidelines for facing the future challenges that we might face regarding ethics and law because of the robotics revolution.

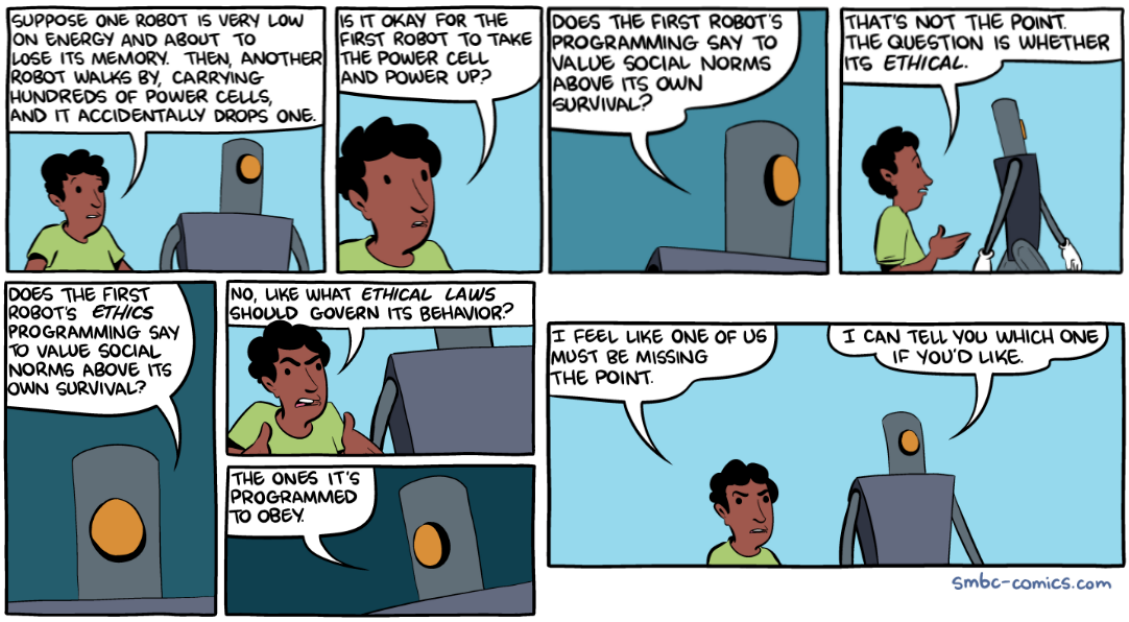

Figure 8.1: Comic presenting a simple example of an ethical debate raised by robotics.

8.1 Roboethics

The term roboethics was introduced at the First International Symposium on Roboethics in 2004 by G. Veruggio [121]. Roboethics is a field of applied ethics aimed at developing tools for the advancement of robotics, studying both the positive and negative implications of robotics for individuals and society, with a view to inspire the moral design, development, and use of intelligent robots, and help prevent their misuse against humankind. In recent years researchers from areas as diverse as robotics, computer science, psychology, law, philosophy, and others are approaching the pressing ethical questions about developing and deploying robotic technology in societies. Fundamental issues in roboethics include the dual-use problem of robots (robots can be used or misused), the anthropomorphization of robots (the illusion that machines have internal states that correspond to the emotions they express, like a "ghost in the machine"), and the equal accessibility to technology challenges, such as for care robots [122]. While military robots were initially a main focus of the discussion, social robots are now becoming an increasingly important topic as well. The prevalence of service robots and their successful cooperation with humans will depend heavily on the extent to which they are accepted by the people who are to work or collaborate with them [123].

One of the first publication directly addressing and setting the foundation for robot ethics was Runaround, a science fiction short story written by Isaac Asimov in 1942 which featured his well known Three Laws of Robotics:

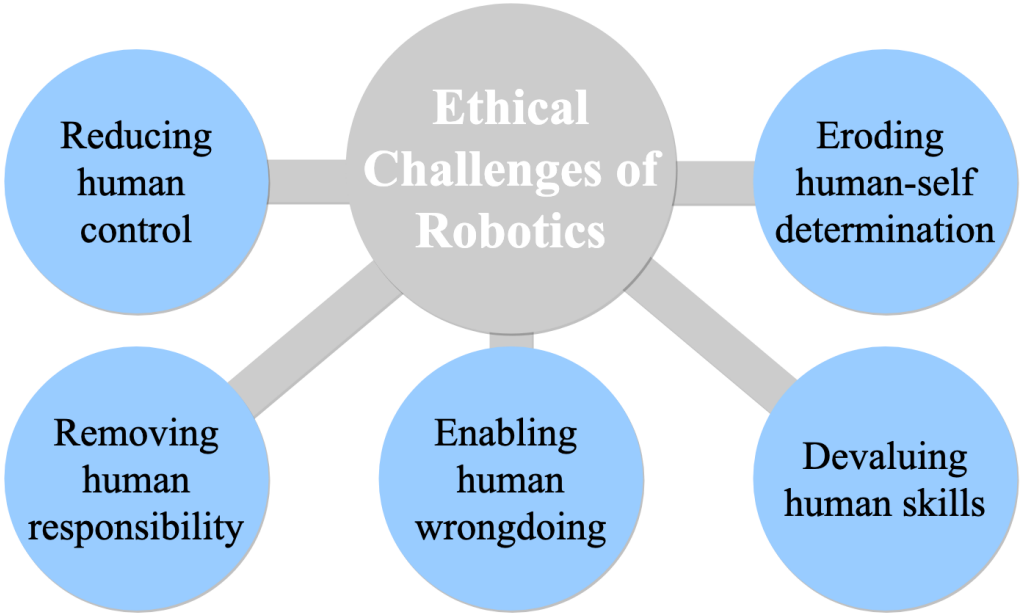

Figure 8.2: Ethical issues of robotics developments (after [124]).

- A robot may not injure a human being or, through inaction, allow a human being to come to harm.
- A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.
- A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

The laws are prioritized to minimize conflicts. However, in story after story, Asimov demonstrated that three simple, hierarchically arranged rules could lead to deadlocks when, for example, the robot received conflicting instructions from two people, or when protecting one person might cause harm to others. It became clear that there are many complex open questions when mixing robotics and ethics.

Robotics and AI pose five fundamental concerns [124], summarized in Figure 8.2. First, excessive reliance on intelligent robots may lead to the delegation of sensitive tasks to autonomous systems that should remain subject to human supervision, either 'in the loop', for monitoring purposes, or 'post-loop' for correcting the robot's actions. Second, robotics and AI may de-responsibilize people whenever an autonomous system could be blamed for a failure. Third, unemployment is an ethical problem, not just an economic one. Intelligent robotics could change the workforce structure, cause a shift in the skills base, and potentially facilitate a complete de-skilling of the workforce even in safety-critical contexts. Fourth, AI may erode human freedom, because it may lead to unplanned and unwelcome changes in human

behaviors to accommodate the routines that make automation work and people's lives easier. Five, it is possible a straightforward AI misuse. This is a problem related to the unethical application of AI by those who control it. AI should be designed and used to treat every human being always as an end and never only as a means.

For addressing the aforementioned issues to some extent, it is fundamental that intelligent robotics incorporate the principle of transparency, which requires that the basic elements of decisions that can have a substantial impact on people can always be provided and that the calculations of the system of AI can be reduced to a form understandable to human beings.

Another very basic principle in the ethical dilemma is that robots cannot do all the work and 'sideline' humans. If this was the case, human conditions would worsen, their learning and cognitive capacities would be modified, and even conditioned to perform tasks like robots, faced with the need to compete with robots as a workforce. Then, advances in the intelligence of robots must be 'driven' by the human society so that robots are used to stimulate humans, increase their capabilities and make them grow as people: 'robots should increase the abilities of people and give them more autonomy, instead of decreasing it' [125].

A general ethical framework governing robot design through to robot use is now needed, composed of a set of ethics applied to robotics and aimed at humankind. In 2016, the European Parliament [126] set the following list of basic roboethical principles for robot users and designers:

1. **Protecting humans from harm caused by robots**: The first principle of roboethics is to protect humans against any harm caused by a robot, for example, in an instance where a technician is operating a health care robot that has just injured patients due to a lack of maintenance or faulty settings.

2. **Respecting the refusal of care by a robot**: This principle follows on from the first and establishes the right for a person to refuse to be cared for by a robot. It may apply in the sensitive situation that arises when a person suffers no physical harm from a robot but feels so profoundly uncomfortable in its presence as to render this presence unbearable.

3. **Protecting human liberty in the face of robots**: This roboethical principle calls for respect of human liberty when using a robot. Some autonomous robots might trample all over freedoms, on the pretext of protecting a person, leading to a clash of certain basic rights, such as protecting liberty versus considering people's health and safety. For example, a security robot might restrain a person who has broken into a shop, or a robot might detain a runaway child at home.

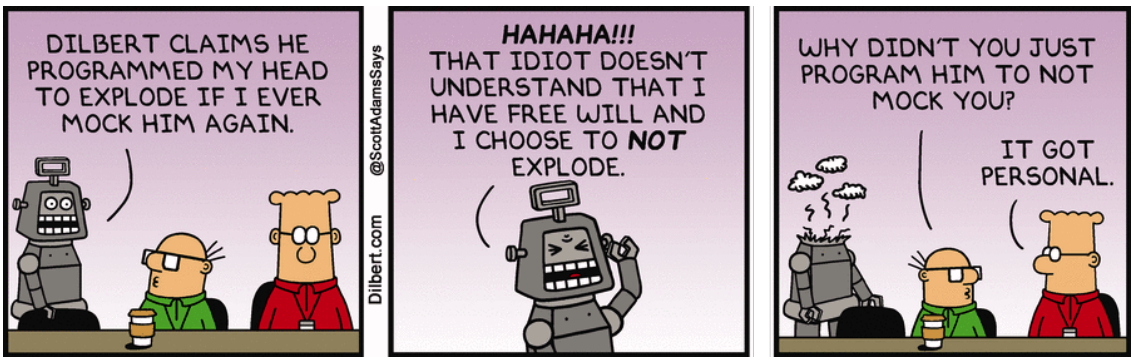

Figure 8.3: Dilbert comic about robots' free will (after Scott Adams).

4. **Protecting humans against privacy breaches committed by a robot**: The aim of this roboethical principle is to protect people from any privacy breaches committed by a robot. The specific nature of the breach would mean that the robot user might see not only their private life exposed but also that of third parties, such as family members, friends, or helpers. A protocol needs to be established, to allow the interested party to remain in control of when a third party is permitted to enter his private sphere.

5. **Managing personal data processed by robots**: Autonomous robots will gather large volumes of data using their various sensors. This would mean that the robot user should always be empowered to prevent the machine from gathering or processing personal data.

6. **Protecting humanity against risk of manipulation by robots**: Emotional robots have some clear advantages when it comes to facilitating interaction between people and robots, as some humanlike robots seek to do with children suffering from autism. Whenever we enable a robot to simulate emotions, there is a risk of a person developing the same type of bond as with another human being.

7. **Avoiding the dissolution of social ties**: Autonomous robots offer a remedy to various problems linked to aging populations. For example, robots assisting elderly people will allow senior citizens to remain at home, even if they lose their health or independence. Therefore, the public health service could make large savings. However, as it will be less expensive for people to have robots rather than a human helper at home, there is a risk of machines becoming the norm and people the exception.

8.2 Legal Framework

If intelligent robots are able to learn and modify their own behavior means that they are capable of manifesting (or, at least, appearing to manifest) human cognitive processes that are associated with the exercise of free will [127] (Figure 8.3). These processes include making choices, forming intentions, reaching decisions, and giving or withholding consent. Robots are the technology of the future. But the current legal system is incapable of handling them. We, therefore, need to start asking ourselves whether a legislative instrument on these technologies is truly necessary and, if so, what this future text might look like.

The first issue when discussing regulation is that of definitions. One cannot regulate something without first defining it. However, the term robot is technical and encompasses a wide range of applications that have very little in common. For this reason, it is a very complex issue to develop a unitary body of rules applicable to all kinds of robotic applications.

The European Parliament [126] defines a smart autonomous robot as a machine encompassing the following characteristics:

- Acquires autonomy through sensors and/or by exchanging data with its environment (inter-connectivity) and trades and analyses data.
- Is self-learning.
- Has physical support.
- Adapts its behaviors and actions to the environment.

This definition relates solely to smart autonomous, and so not all, robots. However, the cited characteristics warrant further clarification to avoid any uncertainty regarding the scope of the future law.

When considering civil law in robotics, we should disregard the idea of autonomous robots having a legal personality (Figure 8.4), for the idea is as unhelpful as it is inappropriate. Traditionally, when assigning an entity legal personality, we seek to assimilate it to humankind. This is the case with animal rights, with advocates arguing that animals should be assigned a legal personality since some are conscious beings, capable of suffering, etc., and so of feelings that separate them from things. Legal personality is therefore not linked to any regard for the robot's inner being or feelings, avoiding the questionable assumption that the robot is a conscious being. Assigning robots such personality would, then, meet a simple operational objective arising from the need to make robots liable for their actions.

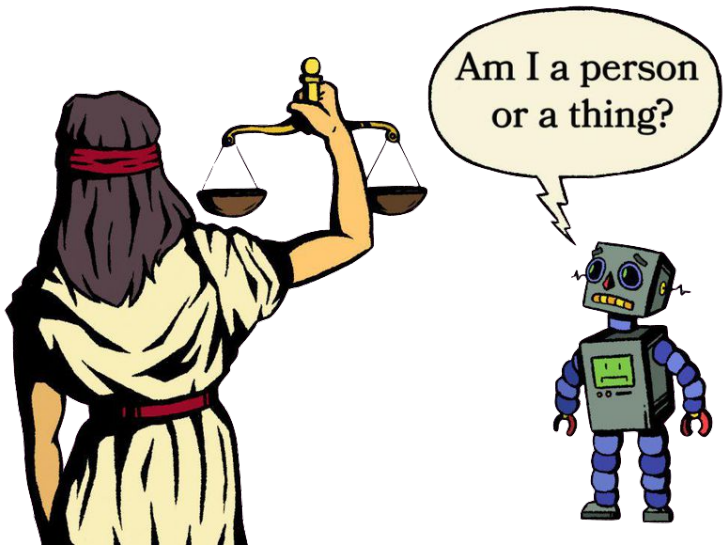

Figure 8.4: A sentient robot asks Lady Justice to be transferred from a legal thing to a legal person for freedoms, rights, and protections (after @MackKennyArt).

We also have to bear in mind that this status would unavoidably trigger unwanted legal consequences. Creating a legal personality would mean that robots' rights and duties had to be respected. How can we contemplate conferring rights and duties on a mere machine? How could a robot have duties, since this idea is closely linked with human morals? Which rights would we bestow upon a robot: the right to life, the right to dignity, the right to equality with humankind, the right to retire, the right to receive remuneration, etc.?

Another fundamental question is the liability for damages caused by an autonomous robot. Conventionally, damage caused by an autonomous robot might arise from a machine defect, for which specific laws have already been developed. These could be applied in several circumstances, particularly if the producer had insufficiently informed the customer of the dangers associated with autonomous robots, or if the robot's safety systems were deficient. Damage caused by autonomous robots might also be traced back to user error. In such instances, either strict or fault-based liability may be imposed, depending on the circumstances.

Nevertheless, autonomous robots will bring about further unprecedented difficulties, since it may be more difficult to ascertain what caused the damage in certain situations, particularly if the robot is able to learn new things by itself. However, it is wrong to say that this calls for new rules which focus on how a machine can be held responsible for its acts or omissions. Bearing in mind the dangers of assuming a robot has a legal personality, it is out of the question that it might be held responsible for its acts or omissions. Only a physical person should be held liable, through various insurance mechanisms.

The European Parliament suggests that the future legislative instrument should provide for the application of strict liability, as a rule, thus requiring only proof that damage has occurred and the establishment of a causal link between the harmful behavior of the robot and the damage suffered by the injured party. In principle, once the ultimately responsible parties have been identified, their liability would be proportionate to the actual level of instructions given to the robot and of its autonomy, so that the greater a robot's learning capacity or autonomy is, the lower other parties' responsibility should be, and the longer a robot's 'education' has lasted, the greater the responsibility of its 'teacher' should be; notes, in particular, that skills resulting from 'education' given to a robot should not be confused with skills depending strictly on its self-learning abilities [126].

The previous leaves many ideas to explore. It must be ensured that the future instrument is more accurate and simpler to implement since one potential concern is that judges who are little-versed in emerging technologies might have trouble comprehending the subtleties. Moreover, we should take into account many other various possible scenarios, such as the one in which a self-learning robot is hired out (a growing trend).

8.3 Managing the Ethical-legal Impact

Service robots are designed to interact with human beings in domestic environments. The wide range of applications made available by the new generation of robots has raised a number of issues that are not solely related to engineering and should be addressed during the design process.

Among the major consequences of coexisting with human beings, besides safety and usability requirements, is the imperative need to guarantee their 'social acceptability'. Acceptability is usually described as the willingness within a user group to employ robots for the task it is designed to support [128]. The social, as well as the functional use of robots, differentiates them from traditional consumer technology appliances. How we view and interact with them; the familiarity, trust, and confidence we might place in them; and our high expectations of how they will work fed by media, film, and our own imagination mean that robotics warrants special consideration when considering how to protect consumers. At the same time, we need to ensure that any fears and risks applicable to the design are seen in context, and balanced with the need to ensure innovators feel on safe ground in developing and marketing the new and exciting robotic applications [129].

Several complex questions regarding ethics and legislation will arise in the context of advanced robotization. We must not look away. Intelligent robotics offer an unimaginable spectrum of possibilities, and it is roboticists' responsibility to get

educated on the potential effects to make correct statements on the consequence of robotization. It is important that society is introduced to the new intelligent service robotics technologies and debates about their implications. This will inevitably lead to adaptations to existing laws and the drafting of new legislation and regulation where there are gaps.

In order to achieve full social acceptability of intelligent service robots, it is essential to take into account that technological progress does not just happen but is driven (at least for now) by human decisions on what, where, and how to innovate. It would be misplaced to succumb to techno-fatalism and view our fate as pre-determined by blind technological forces and market forces that are beyond our control. Instead, our future is shaped jointly by the technological innovations that we humans create, by the social and economic institutions that we collectively design, and by the ethical values that guide it all. We as a society have the power to confront the challenges posed by our technological possibilities and, through individual and collective action, actively steer the path of technological progress in AI so as to shape the future that we want to live in [130].

Chapter 9

Environmental Impact of Robotics

The development of advanced automated systems is expected to offer a high degree of customization of goods and services, and decentralized production systems. This might bring significant changes to current production and consumption patterns, altering the associated demands for energy and resources. We need to consider the environmental footprint caused by the deployment of robotic technologies in order to create improvements compared with the processes used today. The advent of robots has the potential to create a unique momentum for revisiting our understanding of sustainability. A strategical environmental analysis of automation may facilitate such debate, decreasing the expected adverse impacts and enhancing environmental benefits. These ideas can support discussions on wider policy proposals, such as those related to the circular economy.

In this chapter, we discuss the potential environmental impact of robotics. First, in Section 9.1, we perform a strategic environmental assessment for evaluating the potential environmental impact caused by the irruption of robots. Then, in Section 9.2, we present some guidelines for sustainable development, allowing us to face some of the current environmental challenges.

9.1 Strategic Environmental Assessment

A strategic environmental assessment (SEA) is a systematic decision support process, aiming to ensure that environmental and possibly other sustainability aspects are considered effectively in policy, plan, and program making. It allows to look at cumulative environmental effects and appropriately address them at the earliest stage of decision-making alongside economic and social considerations. SEA was created in an attempt to extend environmental impact assessments (EIA) to policies, at a more strategic level [131].

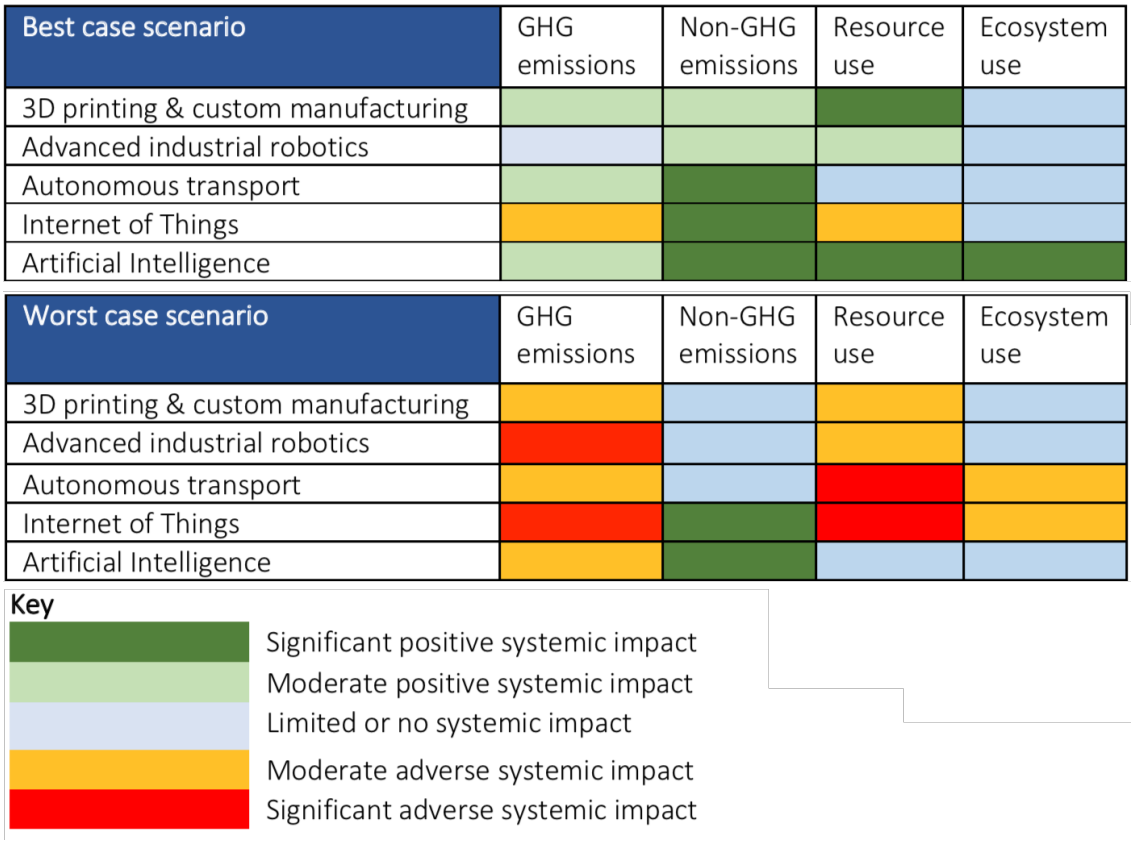

Figure 9.1: Best case and worst-case environmental outcomes of automation (after [132]).

The adoption of automation is typically supported through implicit public policies that are not subject to any formal environmental assessments. In the absence of rigorous studies, it is often suggested that the new technologies will help to solve the most urgent sustainability challenges [133]. However, an emerging body of literature examining the environmental implications of these new technologies suggests that the picture may be mixed. Production and consumption systems created by automation may if carefully managed, deliver environmental improvements compared with the processes used today. Yet if applied without proper environmental considerations and management systems, robots alone may bring adverse impacts.

According to J. Dusik et al. [132], the best-case and worst-case environmental outcomes of automation, in terms of greenhouse gas (GHG) and non-GHG emissions, and the use of resources and the ecosystem, are depicted in Figure 9.1. We can see that regarding the environmental benefits, virtually all automation technologies have the potential to reduce non-GHG emission loads compared with techniques used today. Beneficial impacts can also be achieved in ecosystem uses that are unlikely to be systemically affected by automation and which may even profit from opportunities for improved environmental monitoring, management, and potentially environmental accounting systems.

The picture is much more varied when examining the impacts of automation technologies on resource use. Even if some innovations aim to minimize energy consumption the operation of the entire information and communication system supporting automation is expected to significantly increase the use of electricity and potentially also GHG emissions, depending on the power source used. The proliferation of electronic appliances and equipment will also increase demands for input materials and will generate volumes of electronic waste that will need to be properly managed and ideally recycled.

Regarding the consumption of rare materials in the manufacture of the robots themselves, it must be taken into account that robots are not different from any machine with high precision and high durability parts. Robot manufacturing will require sustainable solutions similar to those for other products. Sustainable materials development for robotics mainly targets two scientific questions. First, can we employ new materials and resources that contribute to a more sustainable future? and second, how can we use or modify existing materials to reduce their ecological footprint on the environment? Solutions to the first question include high-performance materials from renewable sources, or biodegradable ones, with the target to save valuable resources or to reduce waste. The same goals apply for solutions to the second question, but instead of developing new materials, they target fabrication processes, recycling, and product designs [134].

We have to consider the life-cycle of robotics as a whole (Figure 9.2), taking into account the economic and environmental cost of producing, maintaining, and disposing of the robots. For a technology that has to meet high-performance standards, recycling represents a feasible approach toward more sustainable use of technology. In general, robot recycling follows economical viewpoints: a product is more likely to be recycled if recycling is cheaper than the fabrication costs of a new one. Therefore, the ideal recycling process must be cost-effective, technologically easily achievable, integrated with closed production-recycling loops, target valuable materials, and require little energy.

To render robotics sustainable, recycling must be already included in the design phase. A successful recycling scheme requires the individual robotic materials to be easily separable to enable uncomplicated reuse, exchange, and upgrade of robots. Another sustainability approach is to use less material by design. Autonomous robots benefit twofold from lightweight materials/component designs, aiming to first reduce the weight and increase operation time, and second minimize environmental impact by reducing the total amount of waste.

As a concluding remark, one must keep in mind that the actual impacts will obviously vary depending on the actual uptake (speed and scale) of automation, and on the economic, social, environmental, and institutional context in which these technologies get deployed.

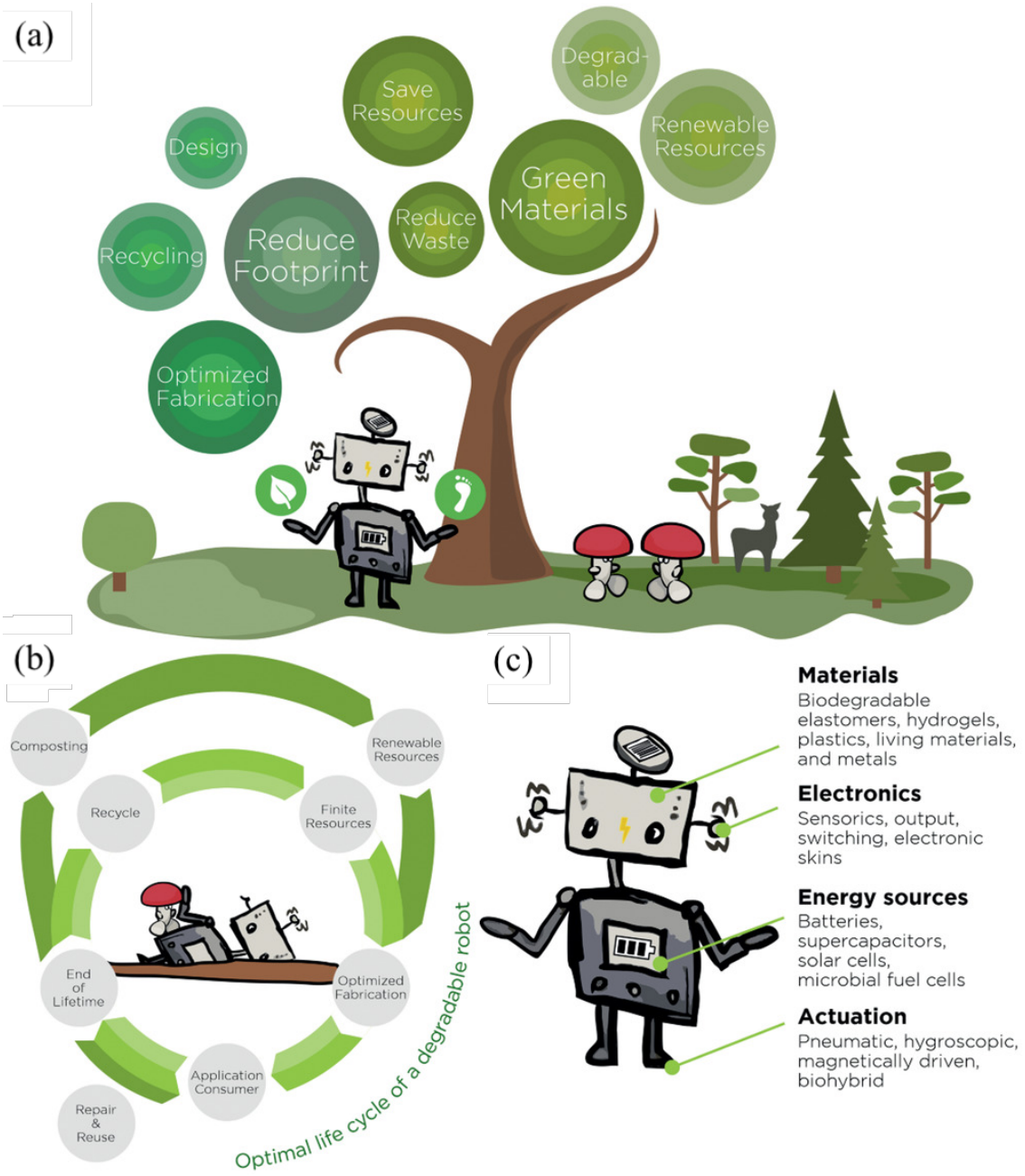

Figure 9.2: Sustainable robotics development. (a) Strategies toward ecofriendly robotics include several pathways from material approaches to optimized design and fabrication. (b) The optimal life-cycle of a robot is a closed-loop, which (c) targets all robotic components from actuation to energy supply (after [134]).

9.2 Managing the Environmental Impact

Robotics and intelligent automation might help communities improve their quality of life and contribute to sustainable development. In the long-term, the adoption of robotics and intelligent automation as part of a development plan must be based upon documented opportunity, feasibility studies, and customized technology solutions designed for the local environment, preferably by local engineers, educators, and policy-makers [135].

Robotics for sustainability could develop from a combination of classical and sustainable technology and new application models. The latter is difficult to imagine and might best be developed by trial and error in the environment where the solutions are used. This relies on creative engineers and entrepreneurs fluent in the local environment who have a working knowledge of the principles of classical robotics and sustainable technology. Utilizing local resources and developing talent is crucial to the successful design and production of applied-technology development solutions. Local technologists and engineers 'have a unique understanding of the relevant problems as well as the cultural context, available resources, strengths and challenges that will influence the creation of innovative and useful solutions' [136].

Chapter 10

Conclusions and Future Work

Endowed with higher levels of autonomy, robots are required to perform increasingly complex manipulation tasks. In this work, we aim at providing a step forward towards enabling robots to learn from human demonstrations. Achieving this would mean opening the doors to robots to operate in assistive domains, and approaching them to people outside fields related to science and engineering. In this final chapter, in Section 10.1, we summarize and discuss the main results from the investigations presented throughout the thesis. We close, in Section 10.2, looking at possible directions for future research.

10.1 Conclusions

We can structure the main conclusions around the following three questions (Section 1.1), which motivated the research of this thesis.

How can a robot imitate the human performance of a task?

We look for the answer to the first question in Chapter 5. In order to transfer the human motion to a robot, we first define a general mapping that allows determining the equivalent robot posture based on human motion capture data. This solves the correspondence problem due to differences in physical embodiment (Section 5.1). However, although we know what the movement of the robot should be, achieving tight coordination between the motion of the robot links poses a challenge from a control perspective. By adopting a whole-body control scheme (Section 5.2), we are able to efficiently compute the required control actions to meet the multiple task constraints simultaneously, achieving a successful imitation in real-time.

Additionally, we have to consider that when robots are operating in human-shared environments, ensuring safety when physical-human robot interaction occurs is of utmost importance. For this purpose, we design a variable admittance controller (Section 5.3), for which we derive the stability conditions, that allows combining a tight position control with a compliant behavior when the robot enters in contact with a human. We validate our approach through several real-world experiments using the mobile robot TIAGo (Section 5.4). The results show an effective real-time imitation and a successful dynamic adaptation of the robot behavior. In light of the obtained performance, we can conclude that this work sets the foundations for providing robots with demonstrations of a task in an intuitive manner.

How can a robot learn and generalize a skill from a few demonstrations?

Seeking a response to the second question, we first reviewed the existing solutions in Chapter 4. The methods proposed in the literature are mainly based on probabilistic nonlinear regression (Chapter 2), being the most advanced and recent one Kernelized Movement Primitives (Section 4.4). This approach is based on a kernelized treatment and can handle high-dimensional learning problems while allowing the adaptation of the motion. Inspired by these findings, we conclude that Gaussian Processes (Chapter 2) have a great potential within the learning from demonstration field as they are the most general and versatile formulation for probabilistic nonlinear regression, and are purely based on a kernel treatment.

In Chapter 6, we develop a novel Gaussian Process-based learning from demonstration framework that encompasses the main components required for a state-of-the-art method. First, we introduce an extended formulation for encoding trajectory distributions by considering multiple outputs, derivative observations, and rotations in the input space (Section 6.1). Then, we address the problem of modeling the latent uncertainty in the demonstrations, which is crucial for reflecting the importance of the different stages of the task (Section 6.2). We resort to Heteroscedastic Gaussian Process models and introduce an algorithm for optimizing their likelihood. Additionally, for adequately capturing the variability in time-invariant policies, we propose the Task Completion Index, which along with the Dynamic Time Warping algorithm allows to correct the time distortions that might appear in the demonstrations performed by a human teacher.

Also, in learning from demonstration is essential to be capable of generalizing the learned motion to a wide range of scenarios. In order to enhance such capability, we introduce a formulation for including task variables in the model, which describe the context and can be either real, integer, or categorical (Section 6.3). For further improving the versatility of the learned policy, we present a simple method for modulating the retrieved motion using via-points, and also an approach for combining several policies into a single representation (Section 6.4).

One of the major limitations of Gaussian Processes is the computational cost associated with the learning process, which limits its applicability for tasks that require a large number of demonstrations. To tackle this problem, we present an alternative formulation that exploits the structure of replications for alleviating the computational complexity (Section 6.5). The main advantage of the proposed approach is that the retrieved model is exact since no approximations or assumptions are made.

Finally, we illustrate the potential of our learning framework through a door opening and a handwriting task (Section 6.6). The results show that the proposed model is capable of generalizing across multiple demonstrations, encode the task uncertainty and adapt to variant task conditions. Therefore, we conclude that Gaussian Processes might be a step forward towards automating complex manipulation tasks.

What is the potential impact of the evolution of robots in our societies?

In order to answer the last question, in Part III, we study the existing literature on the prospective consequences that the development of automation. We start, in Chapter 7, addressing the question from an economic perspective. Mainly, two narratives have emerged. On the one hand, technology pessimists fear that we are headed toward an economic dystopia of extreme inequality and class conflict. On the other hand, technology optimists argue that historically, rapid technological advancements have led to an improvement in human welfare by raising the value of non-automatable tasks. Another major concern is whether robots are going to 'take our jobs' or not. What there is a general consensus about is that most sectors will be affected. The automation of global value chains contributes to the divorce of humans from their labor, from their capacity to earn wages for sustaining human life. Policymakers must begin to consider alternative economic arrangements which do not rely solely on the wage-labor contract at the heart of capitalism.

In Chapter 8, we address the ethical and legal concerns that are becoming more pressing as robotic technologies advance. In recent years, researchers from areas as diverse as robotics, computer science, psychology, law, philosophy, and others are entering the field of roboethics. Some of the general questions involve an excessive reliance on intelligent robots, unemployment, or limitations of human freedoms. Regarding legal frameworks, the major concerns include the definition of a robot, whether it should have a legal personality, and who is responsible for the possible damages that it might cause.

Finally, in Chapter 9, we evaluate the environmental footprint caused by the deployment of robotic technologies. We have to consider the life-cycle of robotics as a whole, taking into account the economic and environmental cost of producing, maintaining, and disposing of the robots. To render robotics sustainable, it is essential to consider the recycling process during the design phase.

As a concluding remark, from the author's viewpoint, it is certain that several complex questions regarding the economy, ethics, law, and the environment will arise along with the deployment of robots in assistive domains. However, intelligent robotics offer an unimaginable spectrum of possibilities, with the appropriate attention and the right policies they open the doors to new sources of value and growth. It is in the hands of scientists and engineers to not look away and anticipate the potential impacts in order to turn robots into the motor of global prosperity.

10.2 Future Work

This thesis provides a step forward towards endowing robots with the capability to learn complex manipulation tasks in assistive and unstructured domains. However, many challenges and avenues for future development still remain to be explored. In this section, we identify and discuss some of the most prominent hurdles in learning from demonstration and suggest potential research directions for overcoming them.

Learning from Multimodal Demonstrations Research suggests that utilizing multimodal information, enhances human learning [137]. Current methods for multimodal learning from demonstrations are limited to a small number of modalities [138]. Approaches that reason over demonstrations in multiple modalities will lead to more robust and adaptive behaviors.

Learning from Multiple Teachers Learning from demonstration methods work well for demonstrations performed by one teacher rather than multiple teachers [139]. Different demonstrators tend to have different priorities and notions of optimal behavior. By considering different levels of expertise we can learn a richer policy. So far, this problem has not been sufficiently addressed.

Transferring the Learned Skills Humans can learn from few demonstrations because they are capable of transferring the knowledge from previously learned tasks to new ones. How can robots leverage past demonstrations of related tasks to quickly learn the current task has been explored in some recent works [140, 141].

Interactive Learning We need learning methods that can estimate the suitability of the taught skill to the current situation. The robot must identify when does it have to request user intervention [142, 143]. By taking a more active role in the learning process, robots' performance could benefit from reasoning about the task from a high-level perspective.

Learning Tasks that Humans Cannot Perform Learning from demonstration methods assume that sample motions of the task are available. However, it might occur that a robot has a physical advantage compared to a human, such as having more than two arms. To achieve a performance beyond human capabilities robots should also be able to learn tasks autonomously.

Bibliography

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Andriy Burkov. *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019.

[3] Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133, 1974.

[4] Hamparsum Bozdogan. Model selection and Akaike's Information Criterion (AIC): The general theory and its analytical extensions. *Psychometrika*, 52(3):345–370, Sep 1987.

[5] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[6] Hildo J. Bilj. *LQG and Gaussian Process Techniques for Fixed-Structure Wind Turbine Control*. Dissertation, Delft University of Technology (TU Delft), The Netherlands, 2018.

[7] Dejan Petelin, Bogdan Filipič, and Juš Kocijan. Optimization of Gaussian Process Models with Evolutionary Algorithms. In *10th International Conference on Adaptive and Natural Computing Algorithms (ICANNGA)*, pages 420–429. Springer, 2011.

[8] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. An Algorithmic Perspective on Imitation Learning. *Foundations and Trends in Robotics*, 7(1-2):1–179, 2018.

[9] Harish Ravichandar, Athanasios Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3, 2020.

[10] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, May 2009.

[11] Aude Billard, Sylvain Calinon, and Rüdiger Dillmann. Learning from humans. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*. Springer, 2016.

[12] Stefan Schaal. Dynamic Movement Primitives-A Framework for Motor Control in Humans and Humanoid Robotics. In H. Kimura, K. Tsuchiya, A. Ishiguro, and H. Witte, editors, *Adaptive Motion of Animals and Machines*, pages 261–280. Springer Tokyo, 2006.

[13] Stefan Schaal. Dynamics Systems vs. Optimal Control: A unifying view. In P. Cisek, T. Drew, and J-F. Kalaska, editors, *Computational Neuroscience: Theoretical Insights into Brain Function*, volume 165 of *Progress in Brain Research*, pages 425–445. Elsevier, 2007.

[14] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, 2013.

[15] Peter Pastor, Mrinal Kalakrishnan, Franziska Meier, Freek Stulp, Jonas Buchli, Evangelos Theodorou, and Stefan Schaal. From dynamic movement primitives to associative skill memories. *Robotics and Autonomous Systems*, 61(4):351–361, 2013.

[16] Matteo Saveriano, Fares J. Abu-Dakka, Aljaz Kramberger, and Luka Peternel. Dynamic Movement Primitives in Robotics: A Tutorial Survey. *ArXiv e-print*, 2021.

[17] Stefan Schaal and Christopher G. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10:2047–2084, 1998.

[18] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning rhythmic movements by demonstration using nonlinear oscillators. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 958–963, 2002.

[19] Roman Weitschat and Harald Aschemann. Safe and Efficient Human–Robot Collaboration, Part II: Optimal Generalized Human-in-the-Loop Real-Time Motion Generation. *IEEE Robotics and Automation Letters*, 3(4):3781–3788, 2018.

[20] Roman Weitschat, Sami Haddadin, Felix Huber, and Alin Albu-Schäffer. Dynamic optimality in real-time: A learning framework for near-optimal robot motions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5636–5643, 2013.

[21] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233–242, 1999.

[22] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 763–768, 2009.

[23] Petar Kormushev, Sylvain Calinon, and Darwin G. Caldwell. Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input. *Advanced Robotics*, 25(5):581–603, 2011.

[24] Luka Peternel, Nikos Tsagarakis, Darwin Caldwell, and Arash Ajoudani. Robot adaptation to human physical fatigue in human–robot co-manipulation. *Autonomous Robots*, 42:1–11, 06 2018.

[25] Clemente Lauretti, Francesca Cordella, Anna Lisa Ciancio, Emilio Trigili, Jose Maria Catalan, Francisco Javier Badesa, Simona Crea, Silvio Marcello Pagliara, Silvia Sterzi, Nicola Vitiello, Nicolas Garcia Aracil, and Loredana Zollo. Learning by demonstration for motion planning of upper-limb exoskeletons. *Frontiers in Neurorobotics*, 12, 2018.

[26] Arnau Carrera, Narcís Palomeras, Natàlia Hurtós, Petar Kormushev, and Marc Carreras. Cognitive system for autonomous underwater intervention. *Pattern Recognition Letters*, 67:91–99, 2015. Cognitive Systems for Knowledge Discovery.

[27] Arash Ajoudani, Andrea Maria Zanchettin, Serena Ivaldi, Alin Albu-Schäffer, Kazuhiro Kosuge, and Oussama Khatib. Progress and prospects of the human-robot collaboration. *Autonomous Robots*, 42(5):957–975, 2018.

[28] Kyoungchul Kong and Doyoung Jeon. Design and control of an exoskeleton for the elderly and patients. *IEEE/ASME Transactions on Mechatronics*, 11(4):428–432, 2006.

[29] Sylvain Calinon. *Continuous extraction of task constraints in a robot programming by demonstration framework*. Dissertation, École Polytechnique Fédérale de Lausanne, Lausanne, 2007.

[30] Geoffrey McLachlan and David Peel. *Finite mixture models*. Wiley Series in Probability and Statistics, 2000.

[31] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4(1):129–145, 1996.

[32] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):286–298, 2007.

[33] Sylvain Calinon, Zhibin Li, Tohid Alizadeh, Nikos G. Tsagarakis, and Darwin G. Caldwell. Statistical dynamical systems for skills acquisition in humanoids. In *12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 323–329, 2012.

[34] MS Windows NT kernel description. https://calinon.ch. Accessed: 2021-06-21.

[35] Sylvain Calinon. A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, 9(1):1–29, 2016.

[36] Andrew Ng, Michael Jordan, and Yair Weiss. On Spectral Clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002.

[37] Radford M. Neal and Geoffrey E. Hinton. A view of the EM Algorithm that justifies Incremental, Sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, page 355–368. MIT Press, 1999.

[38] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum Likelihood from incomplete data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[39] Sylvain Calinon and Aude Billard. A probabilistic programming by demonstration framework handling constraints in joint space and task space. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 367–372, 2008.

[40] Emmanuel Pignat and Sylvain Calinon. Bayesian Gaussian Mixture Model for Robotic Policy Imitation. *IEEE Robotics and Automation Letters*, 4(4):4452–4458, 2019.

[41] Chih-Yi Chiu, Shih-Pin Chao, Ming-Yang Wu, Shi-Nine Yang, and Hsin-Chih Lin. Content-based retrieval for human motion data. *Journal of Visual Communication and Image Representation*, 15(3):446–466, 2004.

[42] Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. Probabilistic Movement Primitives. In *Advances in Neural Information Processing Systems*, volume 26, 01 2013.

[43] Alexandros Paraschos. *Robot skill representation, learning and control with probabilistic movement primitives*. Dissertation, Technische Universität Darmstadt, 2017.

[44] Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. Using probabilistic movement primitives in robotics. *Autonomous Robots*, 42(3):529–551, 2018.

[45] Marc Toussaint. Robot trajectory optimization using approximate inference. In *26th Annual International Conference on Machine Learning*, ICML '09, page 1049–1056, 2009.

[46] Sebastian Gómez-González, Gerhard Neumann, Bernhard Schölkopf, and Jan Peters. Using probabilistic movement primitives for striking movements. In *IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 502–508, 2016.

[47] Guilherme J. Maeda, Gerhard Neumann, Marco Ewerton, Rudolf Lioutikov, Oliver Kroemer, and Jan Peters. Probabilistic movement primitives for coordination of multiple human-robot collaborative tasks. *Autonomous Robots*, 41(3):593 – 612, March 2017.

[48] Thibaut Kulak, Hakan Girgin, Jean-Marc Odobez, and Sylvain Calinon. Active learning of bayesian probabilistic movement primitives. *IEEE Robotics and Automation Letters*, 6(2):2163–2170, 2021.

[49] Yanlong Huang, Leonel Rozo, João Silvério, and Darwin G. Caldwell. Kernelized Movement Primitives. *The International Journal of Robotics Research*, 38(7):833–852, 2019.

[50] Yanlong Huang. robinflib-matlab. https://github.com/yanlongtu/robInfLib-matlab, 2021. Accessed: 2021-06-21.

[51] Fernando Pérez-Cruz. Kullback-Leibler divergence estimation of continuous distributions. In *IEEE International Symposium on Information Theory*, pages 1666–1670, 2008.

[52] Jens Kober, Erhan Öztop, and Jan Peters. Reinforcement Learning to adjust robot movements to new situations. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2650–2655, 2011.

[53] Yanlong Huang, Leonel Rozo, João Silvério, and Darwin G. Caldwell. Nonparametric imitation learning of robot motor skills. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5266–5272, 2019.

[54] João Silvério, Yanlong Huang, Fares J. Abu-Dakka, Leonel Rozo, and Darwin G. Caldwell. Uncertainty-aware imitation learning using kernelized movement primitives. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 90–97, 2019.

[55] Jiatao Ding, Xiaohui Xiao, Nikos Tsagarakis, and Yanlong Huang. Robust gait synthesis combining constrained optimization and imitation learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3473–3480, 2020.

[56] Sunny Katyara, Fanny Ficuciello, Darwin G. Caldwell, Bruno Siciliano, and Fei Chen. Leveraging kernelized synergies on shared subspace for precision grasp and dexterous manipulation. *CoRR*, 2020.

[57] Chrystopher L. Nehaniv and Kerstin Dautenhahn. Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation. *Interdisciplinary Approaches to Robot Learning*, pages 136–161, 2000.

[58] Dana Kulić. Human Motion Imitation. In A. Goswami and P. Vadakkepat, editors, *Humanoid Robotics: A Reference*. Springer, 2018.

[59] Matthew Field, David Stirling, Fazel Naghdy, and Zengxi Pan. Motion capture in robotics review. In *IEEE International Conference on Control and Automation*, pages 1697–1702, 2009.

[60] Luis Sentis and Oussama Khatib. A whole-body control framework for humanoids operating in human environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2641–2648, 2006.

[61] Alexander Dietrich. *Whole-Body Impedance Control of Wheeled Humanoid Robots*. Springer Tracts in Advanced Robotics 116, Springer International, 2016.

[62] Kerstin Dautenhahn and Chrystopher L. Nehaniv, editors. *Imitation in Animals and Artifacts*. MIT Press, Cambridge, MA, USA, 2002.

[63] Yasser Mohammad and Toyoaki Nishida. Tackling the correspondence problem. In T. Yoshida, G. Kou, A. Skowron, J. Cao, H. Hacid, and N. Zhong, editors, *Active Media Technology AMT. Lecture Notes in Computer Science, vol 8210*. Springer, 2013.

[64] Leonel Rozo, Pablo Jiménez, and Carme Torras. A robot learning from demonstration framework to perform force-based manipulation tasks. *Intelligent Service Robotics*, 6:33–51, 2013.

[65] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):172–186, 2021.

[66] Sebastian Starke, Norman Hendrich, Dennis Krupk, and Jianwei Zhang. Evolutionary multi-objective inverse kinematics on highly articulated and humanoid robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6959–6966, 2017.

[67] IEEE RAS Technical Committee. Whole-body control. https://www.ieee-ras.org/whole-body-control, 2019.

[68] Luis Sentis and Federico L. Moro. Whole-body control of humanoid robots. In A. Goswami and P. Vadakkepat, editors, *Humanoid Robotics: A Reference*. Springer, 2018.

[69] Bruno Siciliano and Jean-Jacques Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *Fifth International Conference on Advanced Robotics*, pages 1211–1216, 1991.

[70] Oussama Kanoun, Florent Lamiraux, and Pierre-Brice Wieber. Kinematic Control of Redundant Manipulators: Generalizing the Task-Priority Framework to Inequality Task. *IEEE Transactions on Robotics*, 27(4):785–792, 2011.

[71] Arvid QL. Keemink, Herman van der Kooij, and Arno HA. Stienen. Admittance control for physical human–robot interaction. *The International Journal of Robotics Research*, 37(11):1421–1444, 2018.

[72] Christian Ott and Yoshihiko Nakamura. Admittance Control using a Base Force/Torque Sensor. *IFAC Proceedings Volumes*, 42(16):467–472, 2009. 9th IFAC Symposium on Robot Control.

[73] Klas Kronander and Aude Billard. Stability considerations for variable impedance control. *IEEE Transactions on Robotics*, 32(5):1298–1305, 2016.

[74] Miguel Arduengo, Ana Arduengo, Adrià Colomé, Joan Lobo-Prat, and Carme Torras. Human to Robot Whole-Body Motion Transfer. In *IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, July 2021.

[75] Jordi Pagès, Luca Marchionni, and Francesco Ferro. TIAGo: The modular robot that adapts to different research needs. In *International Workshop on Robot Modularity, IROS*, 2016.

[76] Martin Schepers, Matteo Giuberti, and Giovanni Bellusci. Xsens MVN: Consistent Tracking of Human Motion using Inertial Sensing. Technical report, Xsens Technologies B.V., 2018.

[77] Nicolas Mansard, Olivier Stasse, Paul Evrard, and Abderrahmane Kheddar. A versatile Generalized Inverted Kinematics implementation for collaborative working humanoid robots: The Stack of Tasks. In *International Conference on Advanced Robotics*, 2009.

[78] Mauricio A. Álvarez, Lorenzo Rosasco, and Neil D. Lawrence. Kernels for Vector-Valued Functions: A Review. *Foundations and Trends in Machine Learning*, 4(3):195–266, 2012.

[79] Ercan Solak, Roderick Murray-Smith, William E Leithead, Douglas J. Leith, and Carl-Edward Rasmussen. Derivative observations in Gaussian Process models of dynamic systems. In *15th International Conference on Neural Information Processing Systems (NIPS)*, page 1057–1064, 2003.

[80] Muriel Lang. *Human Motor Behavior Prediction through Gaussian Process Modeling on Manifolds*. Dissertation, Technische Universität München, München, 2019.

[81] Bob Palais and Richard S. Palais. Euler's fixed point theorem: The axis of a rotation. *Journal of Fixed Point Theory and Applications*, 2:215–220, 2007.

[82] Quoc V. Le, Alex J. Smola, and Stéphane Canu. Heteroscedastic Gaussian Process Regression. In *2nd International Conference on Machine Learning (ICML)*, page 489–496, 2005.

[83] Kristian Kersting, Christian Plagemann, Patrick Pfaff, and Wolfram Burgard. Most-likely Heteroscedastic Gaussian Process regression. In *ACM International Conference Proceeding Series*, volume 227, pages 393–400, 2007.

[84] Qiu-Hu Zhang and Yi-Qing Ni. Improved Most Likely Heteroscedastic Gaussian Process Regression via Bayesian Residual Moment Estimator. *IEEE Transactions on Signal Processing*, 68:3450–3460, 2020.

[85] Ibrahim Almosallam. *Heteroscedastic Gaussian Processes for uncertain and incomplete data*. Dissertation, University of Oxford, 2017.

[86] Meinard Müller. Dynamic Time Warping. *Information Retrieval for Music and Motion*, 2:69–84, 01 2007.

[87] Olivier Roustant, Esperan Padonou, Yves Deville, Aloïs Clément, Gillaume Perrin, Jean Giorla, and Henry Wynn. Group kernels for Gaussian Process metamodels with categorical inputs, 2018.

[88] David Duvenaud. *Automatic model construction with Gaussian Processes*. Dissertation, University of Cambridge, 2014.

[89] Mickaël Binois, Jiangeng Huang, Robert B. Gramacy, and Mike Ludkovski. Practical Heteroscedastic Gaussian Process modeling for large simulation experiments. *Journal of Computational and Graphical Statistics*, 27(4):808–821, 2018.

[90] Miguel Arduengo, Adrià Colomé, Joan Lobo-Prat, Luis Sentis, and Carme Torras. Gaussian-Process-based Robot Learning from Demonstration. *ArXiv e-print*, 2020.

[91] Miguel Arduengo, Adrià Colomé, Júlia Borràs, Luis Sentis, and Carme Torras. Task-Adaptive Robot Learning from Demonstration with Gaussian Process Models under Replication. *IEEE Robotics and Automation Letters (RA-L)*, 6(2):966–973, 2021.

[92] Jürgen Sturm. *Approaches to Probabilistic Model Learning for Mobile Manipulation Robots*. Springer Tracts in Advanced Robotics (STAR). Springer, 2013.

[93] Severin Lemaignan, Alexis Jacq, Deanna Hood, Fernando Garcia, Ana Paiva, and Pierre Dillenbourg. Learning by Teaching a Robot: The Case of Handwriting. *IEEE Robotics Automation Magazine*, 23(2):56–66, 2016.

[94] International Federation of Robotics (IFR). *World Robotics: Service Robots*. 2018.

[95] Laura Aymerich-Franch. Why it is time to stop ostracizing social robots. *Nature Machine Intelligence*, 2, 07 2020.

[96] Andrew Berg, Edward F. Buffie, and Luis-Felipe Zanna. Should We Fear the Robot Revolution? (the correct answer is yes). *IMF Working Paper*, 2018.

[97] Richard B. Freeman. Who owns the robots rules the world. *IZA World of Labor*, 2015.

[98] David Autor. Skills, Education, and the Rise of Earnings Inequality among the Other 99 Percent. *Science*, 344:843–851, 2014.

[99] James Manyika, Michael Chui amd Mehdi Miremadi, Jacques Bughin, Katy George, Paul Willmott, and Martin Dewhurst. A Future That Works: Automation, Employment and Productivity. Technical report, McKinsey Global Institute, McKinsey and Company, 2017.

[100] Miguel Arduengo and Luis Sentis. The Robot Economy: Here It Comes. *International Journal of Social Robotics (SORO)*, 2020.

[101] Andy Stalman. *HumanOffOn*. Deusto, 2016.

[102] Michel Servoz. The future of work? Work of the future! Technical report, European Comission (EC), 2019.

[103] Angela Jäger, Cornelius Moll, Oliver Som, Crhristoph Zandker, Steffen Kinkel, and Ralph Lichtner. Analysis of the Impact of robotic systems on employment in the European Union: Update. Technical report, prepared by Fraunhofer ISI for the DG Communications Networks, Content and Technology, European Commission (EC), 12 2016.

[104] Gerlind Wisskirchen, Blandine Thibault Biacabe, Ulrich Bormann, Annemarie Muntz, Gunda Niehaus, Guillermo Jiménez-Soler, and Beatrice von Brauchitsch. Artificial intelligence and robotics and their impact on the workplace. Technical report, IBA Global Employment Institute, 5 2017.

[105] William J. Baumol. Macroeconomics of Unbalanced Growth: The Anatomy of Urban Crisis. *American Economic Review*, pages 415–426, 1967.

[106] Adair Turner. Capitalism in the age of robots: work, income and wealth in the 21st-century. In *Lecture at School of Advanced International Studies*. John Hopkins University, Washington DC, 2018.

[107] Frank Pasquale. *New Laws of Robotics: Defending Human Expertise in the Age of AI*. Belknap Press, 2020.

[108] Daron Acemoglu and Pascual Restrepo. Robots and Jobs: Evidence from US Labor Markets. *Journal of Political Economy*, 128(6):2188–2244, 2020.

[109] Jay Dixon, Bryan Hong, and Lynn Wu. The Robot Revolution: Managerial and Employment Consequences for Firms. Technical report, NYU Stern School of Business, June 2018.

[110] Gaaitzen J. de Vries, Elisabetta Gentile, Sebastien Miroudot, and Konstantin M. Wacker. The rise of robots and the fall of routine jobs. *Labour Economics*, 66:101885, 2020.

[111] Andrea Gentili, Fabiano Compagnucci, Mauro Gallegati, and Enzo Valentini. Are machines stealing our jobs? *Cambridge Journal of Regions, Economy and Society*, 13(1):153–173, 2020.

[112] Stanislav Ivanov. Robonomics: Principles, Benefits, Challenges, Solutions. *Yearbook of Varna University of Management*, 10:283–293, 2017.

[113] Pavlina R. Tcherneva. *The Case for a Job Guarantee*. Polity Press, 2020.

[114] Stephanie Kelton. *The Deficit Myth*. Public Affairs, 2020.

[115] William Mitchell, Larry Randall Wray, and Martin Watts. *Macroeconomics*. Red Globe Press, 2019.

[116] Robert Skidelsky. The Case for a Guaranteed Job. *Project Syndicate, Aug 16*, 2019.

[117] Rebecca Butler, Zoe Pruitt, and Eva Wiese. The effect of social context on the mind perception of robots. *Proceedings of Human Factors and Ergonomics Society Annual Meeting*, 63(1):230–234, 2019.

[118] Patrick Lin, Keith Abney, and George A. Bekey. *Robot Ethics: The Ethical and Social Implications of Robotics*. The MIT Press, 2011.

[119] Carme Torras. Service robots for citizens of the future. *European Review*, 24(1):17–30, 2016.

[120] Ryan Calo, A. Michael Froomkin, and Ian Kerr. *Robot Law*. Edward Elgar Publishing, Incorporated, 2016.

[121] Gianmarco Veruggio and Fiorella Operto. Roboethics: Social and ethical implications of robotics. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics*, pages 1499–1524. Springer-Verlag, 2008.

[122] Mika Westerlund. An ethical framework for smart robots. *Technology Innovation Management Review*, 10(1):35–44, 2020.

[123] Julian Stubbe, Johannes Mock, and Steffen Wischmann. The Acceptance of Service Robots: Tools and Strategies for the Successful Development in Companies. Working paper, PAiCE Program, Federal Ministry of Economics and Energy (BMWi), Germany, 2019.

[124] Guang-Zhong Yang, Jim Bellingham, Pierre E. Dupont, Peer Fischer, Luciano Floridi, Robert Full, Neil Jacobstein, Vijay Kumar, Marcia McNutt, Robert Merrifield, Bradley J. Nelson, Brian Scassellati, Mariarosaria Taddeo, Russell Taylor, Manuela Veloso, Zhong Lin Wang, and Robert Wood. The grand challenges of science robotics. *Science Robotics*, 3(14), 2018.

[125] Carme Torras. *The Vestigial Heart: A Novel of the Robot Age*. The MIT Press, 2018.

[126] Nathalie Nevejans. European Civil Law Rules in Robotics. Technical report, Policy Departament of Citizens' Rights and Constitutional Affairs, Directorate-General for Internal Policies, European Parliament's Legal Affairs Committee, 2016.

[127] Tom Allen and Robin Widdison. Can Computers Make Contracts? *Harvard Journal of Law and Technology*, 9(1):25–42, 1996.

[128] Pericle Salvini, Cecilia Laschi, and Paolo Dario. Design for Acceptability: Improving Robots' Coexistence in Human Society. *International Journal of Social Robotics*, 2(4):451–460, 2010.

[129] Chris Holder, Vikram Khurana, Faye Harrison, and Louisa Jacobs. Robotics and Law: Key legal and regulatory implications of the robotics age (Part I of II). *Computer Law and Security Review*, 32(3):383–402, 2016.

[130] Anton Korinek. Integrating ethical values and economic value to steer progress in artificial intelligence. Working Paper 26130, National Bureau of Economic Research, August 2019.

[131] Christopher Wood and Mohammed Dejeddour. Strategic Environmental Assesment: EA of policies, plans and programmes. *Impact Assessment*, 10(1):3–22, 1992.

[132] Jiri Dusik. Strategic Environmental and Social Assessment of Automation: Scoping Summary. Technical report, 07 2018.

[133] PwC. Fourth Industrial Revolution for the Earth. Harnessing Artificial Intelligence for the Earth. Technical report, PwC report for World Economic Forum, Davos, January 2018.

[134] Florian Hartmann, Melanie Baumgartner, and Martin Kaltenbrunner. Becoming Sustainable: The New Frontier in Soft Robotics. *Advanced Materials*, 2020.

[135] Guido Bugmann, Mel Siegel, and Rachel Burcin. A role for robotics in sustainable development? In *IEEE Conference AFRICON*, pages 1–4, Zambia, September 2011.

[136] G. Ayorkor Mills-Tettey, M. Bernardine Dias, Brett Browning, and Nathan Amanquah. Teaching technical creativity through Robotics: A case study in Ghana. Technical report, Carnegie Melon Robotics Institute, 2007.

[137] Ladan Shams and Aaron R. Seitz. Benefits of multisensory learning. *Trends in Cognitive Sciences*, 12(11):411–417, 2008.

[138] Jaeyong Sung, Seok Hyun Jin, and Ashutosh Saxena. Robobarista: Object Part based Transfer of Manipulation Trajectories from Crowd-sourcing in 3D Pointclouds. In *International Symposium on Robotics Research (ISRR)*, 2015.

[139] Rui Camacho and Donald Michie. Behavioral Cloning: A Correction. *AI Magazine*, 16, 1995.

[140] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *34th International Conference on Machine Learning*, pages 1126–1135, 2017.

[141] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *1st Annual Conference on Robot Learning*, pages 357–368, 2017.

[142] John E. Laird, Kevin Gluck, John Anderson, Kenneth D. Forbus, Odest Chadwicke Jenkins, Christian Lebiere, Dario Salvucci, Matthias Scheutz, Andrea Thomaz, Greg Trafton, Robert E. Wray, Shiwali Mohan, and James R. Kirk. Interactive task learning. *IEEE Intelligent Systems*, 32(4):6–21, 2017.

[143] Dylan P. Losey and Marcia K. O'Malley. Learning the correct robot trajectory in real-time from physical human interactions. *Journal of Human-Robot Interaction (JHRI)*, 9(1), 2019.